GeriApp

APLICACIÓN ANDROID PARA GESTIÓN DE GERIÁTRICOS

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías de Telecomunicación

Universidad Carlos III de Madrid

Alumno: Luis Matías Rodríguez Frías.
Tutora: Florina Almenares Mendoza

UNIVERSIDAD CARLOS III DE MADRID | MADRID, 1 DE JUNIO DE 2015

"This is your last chance. After this, there is no turning back. You take the blue pill—the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill—you stay in Wonderland and I show you how deep the rabbit hole goes"

Morpheus (The Matrix)

Agradecimientos

A todas las personas que de alguna forma u otra han contribuido a que este proyecto haya sido posible.

A todos en general y a ninguno en especial.

Resumen

En la actualidad existen un gran número de centros de mayores y el número de personas que residen en ellos es elevado también. Una de las principales preocupaciones de los trabajadores de los geriátricos y de los familiares de los internos es que éstos tengan la mejor calidad de vida y estén atendidos en las mejores condiciones tanto personales como sanitarias posibles.

El presente proyecto tiene como fin el diseño y creación de una aplicación para la automatización y simplificación de las tareas de gestión del cuidado de los internos de una residencia de ancianos. Gracias al uso de una tableta con Sistema Operativo (SO) Android y dicha aplicación, las enfermeras y personal de atención del centro geriátrico serán capaces de tener un control sanitario y de las actividades diarias de la totalidad de los internos.

La aplicación permite al personal encargado del cuidado de los mayores acceder y modificar la información personal y médica de los pacientes y a su vez gestionar las tareas propias de las labores del día a día.

En el proyecto se identifican claramente tres módulos bien diferenciados:

- Diseño e implementación de una aplicación Android que será usada por el personal del centro geriátrico.
- Diseño y creación de la base de datos en la que se encuentra alojada toda la información de los usuarios registrados en la residencia.
- Integración y establecimiento de servicios web que permitan la comunicación entre la aplicación y la base de datos.

Ha de saberse que la aplicación se ha creado para tabletas de tamaño medio (de 7" a 12") con versión mínima de Android 4.0 (debido a librerías importadas) y que ofrezcan soporten Near Field Communication (NFC). El tamaño es el que se ha considerado adecuado para las labores de toma y visualización de datos.

Abstract

At present there are a large number of Senior Centers and the number of people living in them is too high. One of the main concerns of the geriatrics workers and inmates' relatives is that they have the best quality of life and are therefore treated they in the best possible health and personal conditions.

This project aims to designing and creating an application for automating and simplifying management tasks at care of inmates in a geriatric. By using a tablet with Operating System (OS) Android and the application, the nurses and other geriatric staff will be able to check the inmate's health and the control of their daily activities.

The application allows the personnel in charge of caring for the elderly to access and modify personal and medical information of patients and manage the work of each turn controlling the daily tasks.

Inside the project three distinct modules are identified clearly:

- Design and implementation of an Android application that will be used by nursing home staff.
- Design and creation of the database in which it is stored all the information of registered users at the residence.
- Integration and development of web services that enable communication between the application and the database.

It should be known that the application is built for medium-sized tablets (7 "to 12") with minimum version of Android 4.0 (due to imported libraries) and offer support of Near Field Communication (NFC). The size is what is considered appropriate for taking and viewing data.

Aclaraciones previas

Tal y como dicta la nueva normativa universitaria respecto a Trabajos de Fin de Grado, parte de la memoria ha sido redactada en inglés.

"Los alumnos que estén cursando Grados del Plan 2011 (y el Grado en Tecnologías de Telecomunicación) deberán redactar en inglés de forma obligatoria en la memoria del TFG un resumen extendido (5-10 páginas), los capítulos de introducción y conclusiones (siendo recomendable la escritura de la memoria completa en inglés) para cumplir con los objetivos de aprendizaje marcados en la ficha de la asignatura TFG."

La totalidad de acrónimos y siglas usadas cuyo conocimiento debe ser explicado se encuentran anexados al final del documento.

Índice de contenidos

Índice de tablas

Índice de figuras

1 Introduction

In this chapter the most important terms related to the project context and key aspects, such as motivation, objectives and scope are explained.

1.1 Motivation

It is a proven fact that European and particularly the Spanish population are aging as a result of increased life expectancy. For this reason it has been established many companies dedicated to the care of the elderly sector. This, plus a new business, involves a concern on the part of family members who are in the need of having to hospitalize their loved ones.

The following chart shows how in 2025 it is estimated that the population over 50 years will exceed 45% of the total, and this sector it is the most likely to be admitted to a nursing home.

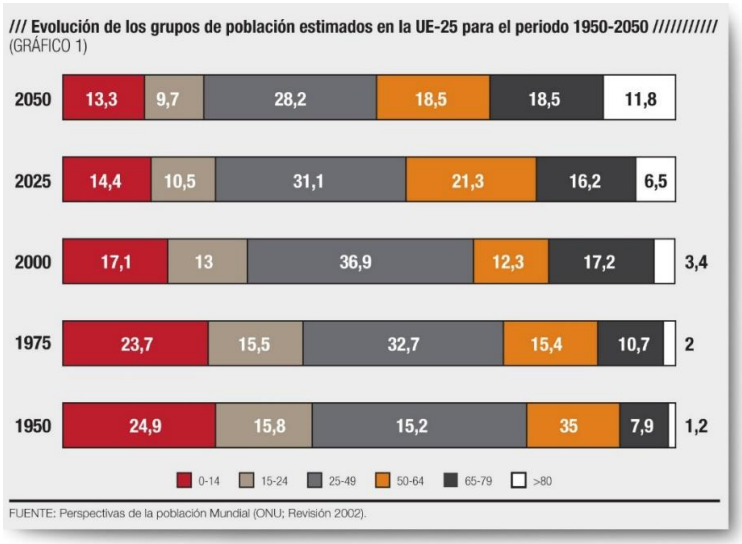


Figura 1: Global evolution of population for the period 1950-2050
http://www.fgcsic.es/lychnos/es_es/articulos/envejecimiento_poblacion

The number of nursing homes is high, as well as the number of people whose age, disability or needs are forced them to live in. The staff working in these centers is responsible of the health and welfare of a large number of patients. With the development of the application GeriApp try to cover the technological shortcomings that can be seen in the sector and facilitate the work of employees, achieving greater efficiency and reducing the number of errors and shortcomings in the conduct of the characteristics of a nursing tasks.

Some of the problems to be solved with GeriApp are:

- Incorrect administration of medication doses.
- Duplication at the administration of medications.
- Lack of communication between center workers when reporting on the tasks that have been made or not.
- Lack of a system control of the work that have not been finished by correspondent turn.
- Depersonalization to treat the interns as numbers without knowing your medical history.
- Ignorance of the particular needs of each user, such as special diets or diseases that require special treatment.
- Lack of a simple mechanism to control internal vital signs visually.
- Excessive paperwork.
- Lack of data centralization.

The idea of computerizing the management of patient information (mainly medical) comes from the study of how current work in most major centers.

Previous to the start of development of the application in question, a brief market survey was conducted, looking for the existence of similar, complementary or supplementary products to which it is intended to develop. As a result of this analysis it was concluded that the commercial software that exists exclusively for this very specific sector is focused on the management of administrative and economic issues. Anyone does not get to control the medical information of patients optimally and provides easy access to workers. In addition, the proposed solution provides added value of portability, providing users GeriApp total freedom to take action, and information management values at the place when dealing with nursing home residents.

While it is true that we are in a boom of technologies which are poorly established by little concepts as the "Internet of things", automation process and interconnectivity between different technologies and devices, it has been note that the staff who will use the application for daily work does not need to have advanced knowledge of computers or electronic systems management. For these reasons, GeriApp uses a friendly GUI application designed for the user and a learning curve as quick as possible so that the process of assimilation of the principles of use is as short as possible.

1.2 Aims

The primary objective when designing and developing the system has been to provide staff of senior centers, nurses and other staff, an efficient tool, simple and portable that allows them to do their job optimizing time, improving patient care and avoiding as far as possible human errors.

Thanks to the three developed modules (application server, connection and database) GeriApp user has the ability to:

- Manage the database that contains the information on internal tasks and specific aspects of the field.
- Check the remaining tasks still to be made on the spot and mark them as done.
- Check the tasks that have already been made to check the proper functioning of rounds.
- Add a task to other employees that can refer to at all times.
- Access database registered in both their personal data such as medical users.
- Search tab for a given user by name or by the number of bed occupied.
- Communicate with the contact persons via email or by telephone.
- Check the history of the most important user's vital signs and see the evolution of the same graphically.

NFC connectivity is an emerging technology that provides easy access to data. It provides a quick way to check the record of individual patient without having to make a written search. The users just have to touch the tablet with the NFC chip for each patient (bracelet, necklace ...) to start working with him.

1.3 Project scope

GeriApp focuses on improving the management of the database of geriatric. It has chosen a mobile support to optimize the tasks of their own rounds in these centers and using NFC technology has been fitted with a plus of simplicity when conducting searches for the records of the patients.

The application is not intended to solve all the daily tasks that are performed in a center for older people rather be a reminder tool to simplify them and to take action and watch users vital signs, like offer a quick and intuitive way to show the medical history of each user giving a more personalized service.

1.4 Dissertation structure

The current project is organized into a total of seven different chapters and seven explanatory annexes.

The **first chapter** is completely descriptive and it presents an overview of the project, explaining what motivated the creation of the system described, to meet its objectives and scope.

The **second chapter** begins with an analysis of the current market in which it is intended to establish the system, exposing both mobile and fixed existing solutions and analyzing their pros and cons. After defining an initial architecture and make the corresponding decisions, a technological study about these two areas of the project is justified. The chapter ends with the description of the use cases and the catalog of software requirements, which allows multiple possibilities offered by the GeriApp system.

In **chapter number three** the code and the internal design of the system modules are described more deeply.

The **fourth chapter** describes the implementation of GeriApp functions in code and tests that have been carried out to check the proper functioning of the overall system.

The legal implications of the project and the legal aspects of its use entails responsibility established in **chapter number five**.

Chapter six describes how the time is organized for the project, as well as physical resources used on it.

The **seventh chapter** is responsible for the conclusions drawn after the PFG and the set out of difficulties and achievements and future work that could lead to GeriApp.

At the end there is a **glossary** in which concisely are defined the acronyms used in the document that are necessary to be known for total understanding.

Following this, it's a list of **bibliography** and references used to access information throughout the entire project.

Finally, a series of explanatory **annexes** had been included to describe manuals and software projects. These are:

- User's manual.
- Programmer's guide.
- Apache-Tomcat server configuration.
- MySQL data base.
- Eclipse Project.
- Android Studio.
- Aplication permissions.

2 Análisis

En este capítulo se analizarán la totalidad de los aspectos del sistema que se ha realizado, describiendo los elementos del mismo.

2.1 Estado del arte

Antes de comenzar a desarrollar cualquier proyecto, es estrictamente necesario analizar el mercado existente. De este modo, en este apartado se analizan las soluciones comerciarles existentes en el campo de la gestión de geriátricos y residencias de mayores.

2.1.1 Programas de PC

Al igual que para cualquier otro tipo de empresa privada o pública, existen variados (aunque no numerosos) programas para la gestión de geriátricos. Al realizar una pequeña investigación se observó cómo muchos de ellos se centraban en aspectos meramente organizativos y económicos, dejando a un lado el control de la información del estado de salud de los pacientes.

A continuación, se detallan las soluciones informáticas más interesantes, populares o accesibles que se encontraron.

Geriges

Se trata de un software bastante completo que aporta la capacidad de gestionar tres grandes ámbitos de un geriátrico:

- La información personal de los pacientes.
- La información de los empleados.
- La gestión económica del centro.

Las siguientes imágenes ejemplifican cómo se ve la interfaz en PC de dicha aplicación comercial. La primera demuestra la vista de la ficha de un interno, la segunda la interfaz de organización de turnos de los empleados y la tercera muestra cómo se presentan los informes económicos y de rentabilidad del centro.

Figura 2: Ejemplos de la interfaz de Gerosalus

Resiges

Software completo e intuitivo que abarca un mayor número de ámbitos y que permite, mediante un sencilla interfaz de PC, gestionar los siguientes campos:

- Residentes.
- Salud.
- Habitaciones.
- Informes.
- Gestión.
- Personal.
- Utilidades.
- Bajas.
- Juzgado.
- Lista de espera.
- Usuario.

La siguiente figura muestra las diferentes opciones que ofrece la aplicación.

Figura 3: Interfaz principal de Resiges 6.0

Geriges

A priori el software más versátil y moderno, al tratarse de una aplicación multiplataforma 100% basada en la nube.

Dispone de un módulo para su uso con pantallas táctiles y compatibilidad con los sistemas operativos actuales para plataformas móviles.

Figura 4: Ejemplo del control del tareas

Como se puede ver en la figura 4, uno de los puntos fuertes de esta aplicación reside en poseer una interfaz clara, completa e intuitiva para la gestión de las tareas del centro, existiendo la opción de crear filtros y asignar tareas a un determinado trabajador.

2.1.2 Aplicaciones móviles

En el momento en que se comenzó a gestar la idea de realizar GeriApp no existían aplicaciones móviles para ninguno de los sistemas operativos mayoritarios que ofreciesen un servicio de valor.

A continuación, se referencian algunas de las aplicaciones que intentan solucionar los problemas a los que se enfrenta la desarrollada en este PFG.

Residencias Sanyres

Aplicación para iOS y Android que se centra en la comunicación del personal del centro geriátrico con los familiares para tenerlos al día de la información sobre la persona que se encuentra enferma.

No aporta más valor que la información sobre sus centros propios.

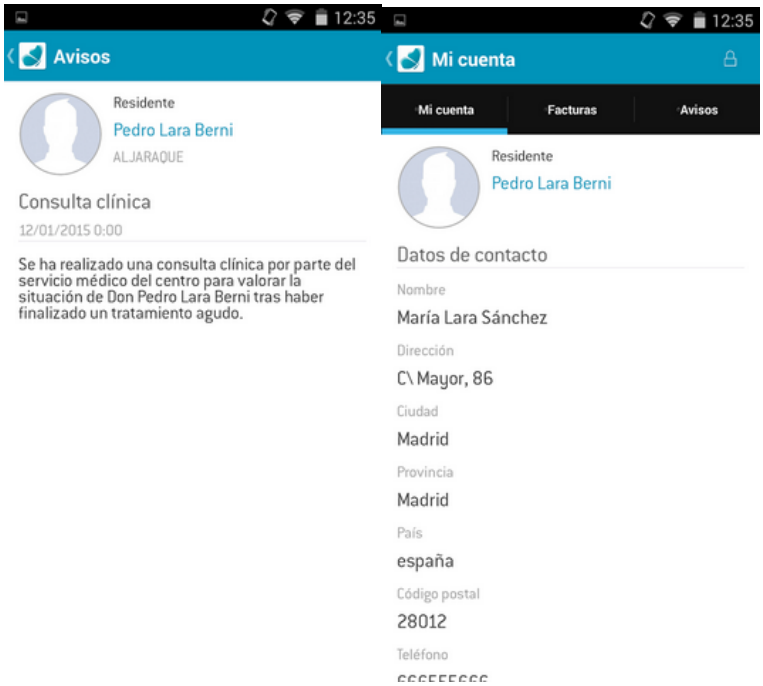


Figura 5: Capturas de pantalla de la aplicación Sanyres

Koontigo Geriátricos

Proporciona una interpretación para los familiares de los informes automatizados que se generan desde el centro de mayores. Establece una comunicación directa y sin costes, con alertas informativas a los familiares a través de mails o SMS.

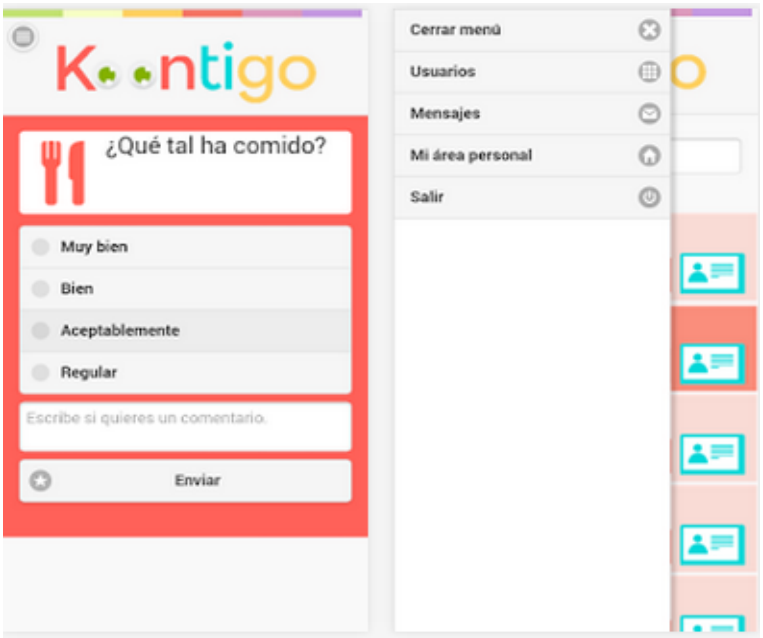


Figura 6: Captura de pantalla de la aplicación Koontigo

2.1.3 Conclusiones

Las características, bondades y deficiencias encontradas en las aplicaciones analizadas se reflejan en la siguiente tabla:

Nombre de la app	Portátil	Gestión de plantilla	Fichas de internos	Creación de informes	Plataforma	Envío de mensajes	Información médica detallada
Gerosalus	No	Sí	Sí	No	PC	Entre trabajadores.	Medicación.
Resiges	No	Sí	Sí	Sí	PC	Para trabajadores y personas de contacto	Sí
Geriges	Sí	Sí	Sí	Sí	Múltiple. 100% en la nube.	Portal familiar.	No completa.
Sanyres	Sí	No	No	No	iOS y Android.	Sí	No
Koontigo	Sí	No	No	No		Sí	No

Tabla 1: Comparativa de aplicaciones para la gestión de geriátricos

Tras el análisis de estas soluciones y otras tantas que se pudieron encontrar en el estudio previo, se llegaron a las siguientes conclusiones:

- Las soluciones comerciales en su mayoría son meras adaptaciones de programas de gestión de Pequeñas y Medianas Empresas (PYMES) al ámbito de la salud.
- No existe ninguna solución portable específica para tabletas Android.
- Se hace necesario incluir de forma más específica el aspecto médico y el historial de los pacientes, ofreciendo así un sistema más completo.

Por consiguiente, se tomó la decisión de crear una app que cumpliese los siguientes requisitos:

- Especial énfasis en los valores sanitarios de los internos y su historial.
- Portabilidad y funcionamiento inalámbrico, tanto Wi-Fi como NFC.
- Uso de una base de datos centralizada y de un servidor único.

2.2 Arquitectura preliminar

A continuación, se exponen de forma gráfica los módulos que forman parte del proyecto, siendo estos los encargados de lograr el nexo de unión entre las acciones tomadas por el usuario desde la interfaz y las acciones derivadas de estas en la base de datos de la aplicación.

Se ha elegido una arquitectura cliente-servidor, puesto que se precisa un alto grado de movilidad y centralización de los datos, ya que GeriApp se ha diseñado para trabajar de forma concurrente con diferentes dispositivos y todos ellos han de tener información coherente y actualizada.

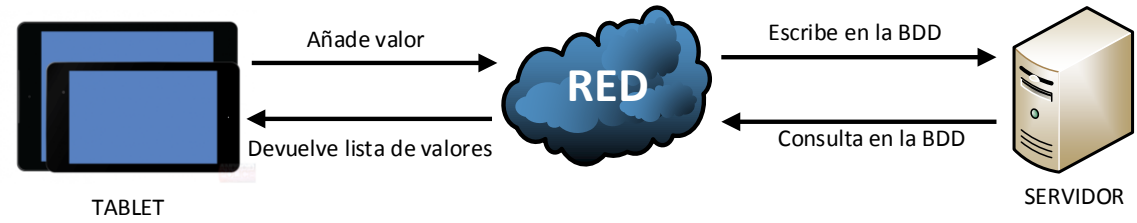


Tabla 2: Arquitectura inicial para el sistema a desarrollar

2.3 Estudio tecnológico

En este apartado se analizarán las diferentes tecnologías que se podrían utilizar a la hora de implementar el sistema, tanto para la aplicación móvil como para el servidor.

Las restricciones impuestas por el diseño inicial del concepto son:

- La compatibilidad con redes Wi-Fi.
- La compatibilidad con la tecnología NFC.
- Funcionamiento con soportes táctiles.

2.3.1 Tecnologías aplicables a la tecnología móvil

A la hora de elegir el SO base para la aplicación móvil GeriApp se estudiaron las siguientes opciones:

- **Android**. Sistema operativo propietario de Google. Que permite fácilmente el uso de Java y XML.
- **iOS**. Sistema operativo de Apple que fue descartado por sus restricciones y requisitos tanto para desarrolladores como usuarios.
- **Windows Phone**. Versión adaptada del SO de Microsoft para dispositivos móviles que debido a su escasa penetración en el mercado de las tabletas no resultaba atractiva.
- **Otros**. Opciones minoritarias tales como Ubuntu Phone, Tizen, Blackberry OS o Firefox OS, cuya elección no era factible debido a su incierto futuro.

2.3.2 Tecnologías aplicables al servidor

A la hora de elegir el tipo de servidor web se plantearon como posibilidades:

- **Apache**.
- **Ngix**.
- **Microsoft IIS** (Internet Information Services).

En cuanto a los lenguajes de programación para el servidor se plantearon como posibles:

- **ASP.NET** (Active Server Pages).
- **PHP** (PHP Hypertext Pre-processor).
- **Java.**

2.3.3 Tecnologías aplicables a la comunicación

Las tres opciones principales que existen para establecer la comunicación entre cliente y servidor son:

- **HTTP** (Hypertext Transfer Protocol). Protocolo extendido y sencillo basado en un modelo petición-respuesta de datos. Se trata como un **web service** que permite la comunicación entre aplicaciones o componentes de estas de diferentes fuentes, sistemas operativos o lenguajes de programación.
- **Sockets + SQL** (Structured Query Language). Mediante los sockets se establece un canal de comunicación capaz de transportar datos de un extremo a otro. Cada socket se encuentra asociado a un puerto para identificar la aplicación destino (por defecto en MySQL es el 3306). Se hace uso de SQL; lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

Para el formato de los datos se estudiaron las siguientes posibilidades:

- **SOAP** (Simple Object Access Protocol). Protocolo que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML
- **REST** (Representational State Transfer). Estilo de arquitectura software para sistemas hipermedia distribuidos.
- **JSON** (JavaScript Object Notation). Formato ligero para el intercambio de datos, se trata de un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

2.3.4 Tecnologías aplicables para el almacenamiento de información

Para el correcto almacenamiento de la información de la aplicación se han elegido dos soluciones: una para la persistencia en el móvil y otra para la gestión de los valores almacenados en la base de datos en el servidor.

Se entiende por Gestor de Base de Datos, la herramienta software capaz de catalogar y almacenar grandes cantidades de información. Entre sus características principales destacan las siguientes:

- Capacidad para añadir, extraer, eliminar o modificar datos.
- Acceso concurrente a los mismos.
- Controla la integridad de la información.
- Recuperación tras fallos en el sistema.

Los gestores más utilizados son los de tipo relacional (RDBMS), en los que la información está organizada en forma de tablas, donde cada fila representa un registro y cada columna representa un campo. Entre las más extendidas se encuentran:

- **PostgreSQL.** Sistema de gestión de bases de datos relacional orientado a objetos y libre.
- **MySQL.** Sistema de gestión de bases de datos relacional, multihilo y multiusuario.
- **Microsoft SQL Server.** Sistema de manejo de bases de datos del modelo relacional, desarrollado por Microsoft.
- **Oracle.** Sistema de gestión de base de datos objeto-relacional muy completo desarrollado por Oracle Corporation.

En cuanto al almacenamiento en el dispositivo móvil Android se estudiaron las siguientes posibilidades:

- Base de datos de tipo **SQLite**.
- **Shared Preferences**.
- Un **fichero de texto plano** en el almacenamiento interno o externo del terminal podría ser empleado en cualquier plataforma.

2.4 Selección de tecnología y arquitectura definitiva de alto nivel

En este apartado se exponen las tecnologías que se han empleado para el diseño y la programación del sistema completo, indicándose la arquitectura de alto nivel del proyecto y explicándose los diferentes componentes que lo conforman. Tras ello, se ilustra el diagrama de despliegue del sistema.

2.4.1 Selección de las tecnologías

En este apartado se justifican las diferentes tecnologías seleccionadas para el desarrollo del proyecto.

Como plataforma de desarrollo para la aplicación móvil se tomó la decisión de adoptar Android en su versión mínima 4.0. Esta decisión se apoya en la diversidad de dispositivos existentes compatibles con dicho SO y la elevada cuota de mercado que posee el sistema operativo de Google.

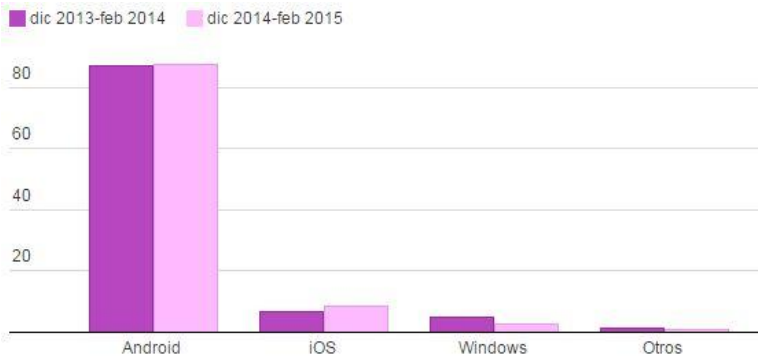


Figura 7: Gráfica de la cuota de mercado de los principales SO para dispositivos móviles

De los dispositivos electrónicos disponibles con Android cuya versión es superior o igual a 4.0, se eligieron aquellos con pantallas de tamaño comprendido entre las 8 y las 11 pulgadas de diagonal y con soporte para NFC.

Para el desarrollo del lado del servidor se tomó Java por ser una tecnología ampliamente conocida antes del proyecto. El servidor se implementa usando una extensión de Java denominada servlet para enlazar la vista con el modelo. Dicho modelo contiene las clases que acceden a los datos de la aplicación. En cuanto al servidor web, se decidió emplear Apache-Tomcat 8.0 por su portabilidad e integración en la mayoría de entornos de desarrollo.

Se decidió emplear HTTP con web services para la comunicación entre el cliente y el servidor, a pesar de la ausencia de una fuerte seguridad.

Para el formato de envío de datos complejos entre el servidor y el cliente se eligió JSON, incluyendo mediante las estructuras de etiqueta-valor los atributos de las clases descritas en el modelo y que representan los elementos de la base de datos.

En cuanto al almacenamiento de los datos en el servidor, la alternativa escogida fue MySQL, debido principalmente a su compatibilidad con Java y Android.

Para almacenar la información de las sesiones de usuario en el dispositivo móvil se decidió emplear Shared Preferences. De esta forma se puede adaptar el fichero XML en cada caso almacenado las constantes propias de cada residencia geriátrica.

2.4.2 Arquitectura definitiva general del sistema

En la siguiente figura se expone la arquitectura definitiva del sistema y en ella se pueden distinguir los componentes necesarios para su correcto funcionamiento.

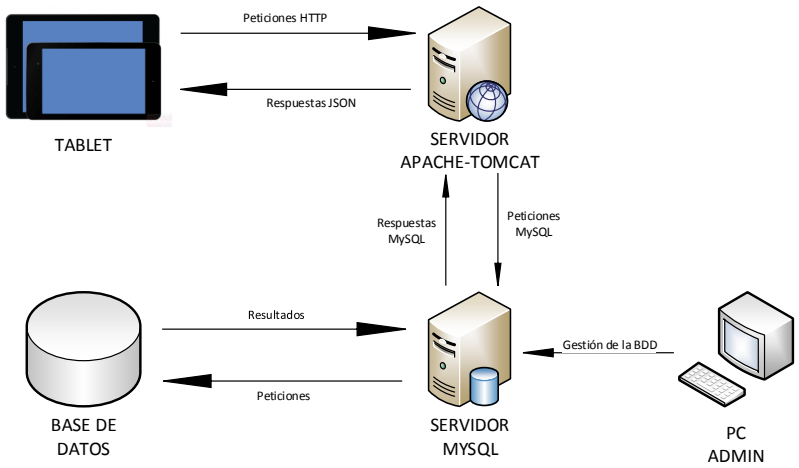


Figura 8: Arquitectura final del sistema completo de GeriApp

A continuación se describen los elementos que forman la estructura global:

- **Tablet**. Dispositivo Android que corre la aplicación GeriApp sirviendo de interfaz gráfica al usuario final. Envía peticiones HTTP y se encarga de interpretar las respuestas obtenidas en formato JSON.
- **Servidor Apache-Tomcat**. Servidor con IP estática montado con Eclipse que se encarga de recibir las peticiones HTTP que genera la aplicación y crear a partir de éstas peticiones MySQL. Las respuestas obtenidas son encapsuladas en un archivo JSON que es devuelto a la tableta.
- **Servidor MySQL**. Servidor para la base de datos instalado en la misma máquina que el servidor Apache. Recibe peticiones y tras ejecutarlas devuelve los resultados para que sean interpretados por el servidor Tomcat.
- **Base de datos**. Almacena la totalidad de la información persistente de la aplicación en diversas tablas.

2.4.3 Diagrama de componentes de alto nivel

En el actual apartado se incluye el diagrama Unified Modeling Language (UML) que modela los componentes de alto nivel, dándose una breve explicación de cada uno de ellos.

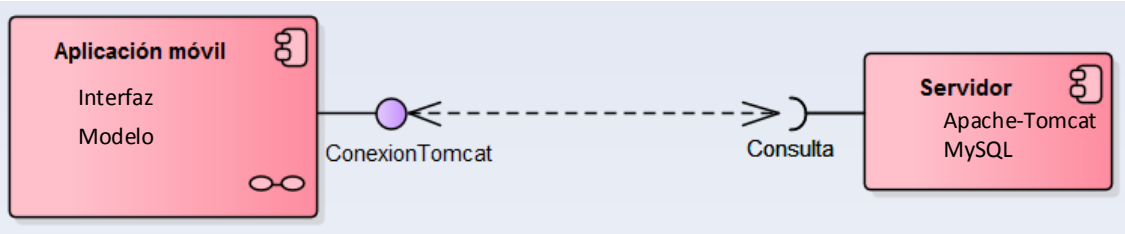


Figura 9: Diagrama de componentes del sistema

2.5 Diagramas de casos de uso

Se entiende por caso de uso la descripción de las funcionalidades de un sistema y las interacciones entre uno o varios actores con éste. De este modo el sistema se define de acuerdo con las necesidades u objetivos los dichos actores.

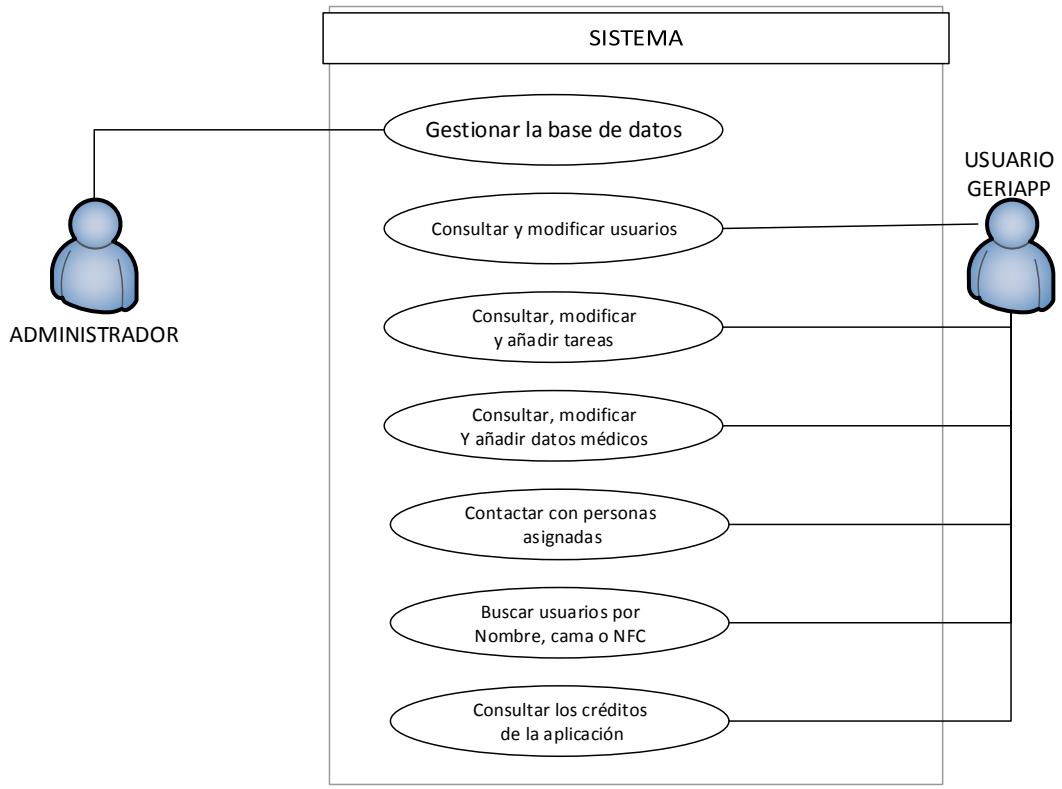


Figura 10: Gráfico explicativo de los actores y casos de uso posibles en el sistema

2.5.1 Actores del sistema

Los actores son los usuarios del sistema que está siendo modelado. En general, se trata de personas, aunque también pueden ser programas o sistemas informáticos. Los diferentes actores que interactúan con el sistema son:

- El usuario **administrador**, que será el encargado de dar de alta a los internos, introduciendo sus datos en la base de datos.
- Los trabajadores (**enfermeros**) del geriátrico, representados en la figura anterior como USUARIO GERIAPP, que son los usuarios finales de la aplicación, que podrán consultar la información de la base de datos centralizada y añadir entradas a la misma.

2.5.2 Formato de la descripción de los casos de uso

Identificador	**Descripción del caso de uso**
Precondición	Condiciones necesarias que se deben dar en el sistema o en el dispositivo.
Objetivos	Propósitos del actor que lleva a cabo el caso de uso.
Secuencia normal	Secuencia de acciones que se dan en el caso de uso.
Secuencia alternativa	Posible escenario alternativo al que se puede llegar desde el caso de uso en cuestión.
Postcondición	Resultado que se cumple tras la realización del caso de uso.

Tabla 3: Modelo del formato de los casos de uso

2.5.3 Caso de uso del Administrador

CU-A01	**Gestionar la base de datos**
Precondición	• El usuario tiene permisos para acceder a la base de datos. • Existe conectividad con el servidor que aloja la base de datos. • Actualmente se debe hacer mediante el IDE MySQL Workbench al carecer el sistema de GUI de administración.
Objetivos	Proporcionar un mecanismo para que desde la administración central de pacientes se puedan dar de alta nuevos usuarios.
Secuencia normal	1. El usuario se identifica como administrador. 2. Si las credenciales lo permiten se accede a la interfaz de gestión. 3. El sistema muestra una pantalla con campos que el usuario deberá cumplimentar. 4. El usuario rellena los campos correspondientes Selecciona la opción de guardar cambios.
Secuencia alternativa	Edición de algún campo para usuarios ya existentes (dar de baja, modificación de datos…)
Postcondición	Almacenamiento del usuario en la base de datos o realizar cualquier otra gestión de la bdd.

Tabla 4: Caso de uso para el administrador

Tal y como se ha mencionado, no se ha creado interfaz para el manejo directo de la base de datos, por lo que el actor Administrador carece de casos de uso dignos de mencionar.

2.5.4 Casos de uso del personal del geriátrico

Principalmente para enfermeras, médicos, celadores y trabajadores encargados del bienestar y salud de los internos.

CU-E01	**Consultar lista de tareas pendientes/hechas**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor se encarga de atender las peticiones.
Objetivos	Proporcionar una vista rápida de las tareas que quedan por realizar o las ya hechas.
Secuencia normal	1. El usuario realiza la consulta. 2. Se envía una petición HTTP para el servidor Apache-Tomcat. 3. El servidor consulta mediante MySQL las tareas y las devuelve a la app. 4. La aplicación interpreta la respuesta y la muestra en pantalla.
Postcondición	Se muestra el listado pedido en pantalla.

Tabla 5: Caso de uso primero para el enfermero

CU-E02	**Añadir tarea para hacer**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor se encarga de atender las peticiones.
Objetivos	Proporcionar la opción de añadir tareas para que las puedan leer el resto de trabajadores.
Secuencia normal	1. El usuario rellena los datos y selecciona guardar. 2. Se envía una petición HTTP para el servidor Apache-Tomcat. 3. El servidor da orden mediante MySQL de almacenar una nueva tarea.
Secuencia alternativa	Si alguno de los datos es erróneo se obtiene un error y se notifica qué valores se deben corregir.
Postcondición	Se almacena la nueva tarea en la base de datos.

Tabla 6: Caso de uso segundo para el enfermero

CU-E03	**Consultar lista de usuarios registrados**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor se encarga de atender las peticiones.
Objetivos	Proporcionar una vista rápida de los internos en la residencia de ancianos.
Secuencia normal	1. El usuario realiza la consulta. 2. Se envía una petición HTTP para el servidor Apache-Tomcat.

	3. El servidor consulta mediante MySQL los usuarios y las devuelve a la app. 4. La aplicación interpreta la respuesta y la muestra en pantalla.
Postcondición	Se muestra el listado pedido en pantalla.

Tabla 7: Caso de uso tercero para el enfermero

CU-E04	**Buscar un usuario determinado**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor que se encarga de atender las peticiones.
Objetivos	Proporcionar la información completa del usuario en cuestión a partir de su número de cama.
Secuencia normal	1. El usuario realiza la consulta. 2. Se envía una petición HTTP para el servidor Apache-Tomcat. 3. El servidor consulta mediante MySQL el usuario. 4. La aplicación interpreta la respuesta y la muestra en pantalla.
Secuencia alternativa	Búsqueda por nombre del interno o por su identificador NFC.
Postcondición	Se muestra la información personal y médica del usuario en pantalla.

Tabla 8: Caso de uso cuarto para el enfermero

CU-E05	**Contactar con la persona asignada**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor se encarga de atender las peticiones.
Objetivos	Permite la comunicación vía email o telefónica.
Secuencia normal	1. El usuario selecciona cómo comunicarse. 2. Se establece la llamada o se crea un mensaje de correo.
Secuencia alternativa	Si el dispositivo carece de la posibilidad de realizar llamadas telefónicas se notifica.
Postcondición	Se lanza el dialer o la aplicación de gestión de correos del dispositivo.

Tabla 9: Caso de uso quinto para el enfermero

CU-E06	**Añadir medidas de constantes**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor que se encarga de atender las peticiones.
Objetivos	Proporcionar una forma de llevar un control del historial sanitario de los internos.
Secuencia normal	1. El usuario toma las medidas. 2. Se anotan en la app. 3. Se envía una petición HTTP para el servidor Apache-Tomcat. 4. El servidor da orden de guardar en la base de datos.
Secuencia alternativa	Se puede modificar uno o varios valores. En el caso

	de no modificar un valor este se mantiene constante.
Postcondición	Se almacena una nueva medida en el historial.

Tabla 10: Caso de uso sexto para el enfermero

CU-E07	**Graficar**
Precondición	• El usuario elige dicha opción. • Existe conectividad con el servidor que se encarga de atender las peticiones. • El interno tiene más de una toma de constantes en la base de datos.
Objetivos	Proporcionar de forma gráfica la información completa de las constantes de un interno.
Secuencia normal	1. El usuario realiza la consulta. 2. Se envía una petición HTTP para el servidor Apache-Tomcat. 3. El servidor consulta mediante MySQL las constantes del usuario. 4. La aplicación interpreta la respuesta y la muestra en pantalla en forma de gráfica lineal.
Secuencia alternativa	
Postcondición	Se muestra el Activity que permite seleccionar que variable graficar.

Tabla 11: Caso de uso séptimo para el enfermero

2. 6 Catálogo de requisitos de software

En este apartado se especifican los requisitos de software. Dichos requisitos se pueden clasificar según su contenido en tres grupos:

- **Funcionales**. Describen una función o tarea específica que el sistema debe ser capaz de llevar a cabo e indican cómo se debe comportar ante entradas o situaciones determinadas.
- **No Funcionales**. Establecen restricciones en el sistema completo. Se clasifican como: de seguridad, privacidad, fiabilidad, usabilidad, disponibilidad y de rendimiento.
- **Inversos**. Delimitan la barrera entre lo que el sistema debe y no debe hacer.

2.6.1 Formato para la especificación de requisitos de software

En este apartado se nombran con el identificador RF-A# los requisitos funcionales del administrador y con el identificador RF-E# aquellos que son propios del equipo de enfermeros, dónde # es un número único de id.

El formato de tabla para los requisitos funcionales y no funcionales es el siguiente:

ID	Nombre	Descripción	Prioridad	Necesidad
ID	Breve descripción del requisito.	Explicación en detalle del requisito.	Prioridad de la implementación del requisito respecto a los demás.	Necesidad para el cliente de este requisito frente a los demás.

Tabla 12: Modelo del formato de tabla para las especificaciones

2.6.2 Requisitos funcionales

ID	Nombre	Descripción	Prioridad	Necesidad
RF-A01	Registro de usuarios	Crear ficha en la base de datos del sistema.	Alta	Alta
RF-E01	Consulta de la información de la app.	Mostrar los créditos de la app así como la información de contacto del creador.	Baja	Media
RF- E02	Consulta tareas pendientes.	Mostrar las tareas pendientes para dar información de que labores quedan por hacer a los trabajadores.	Alta	Alta
RF-E03	Consulta tareas hechas.	Mostrar las tareas ya realizadas para una mejor organización.	Media	Media
RF-E04	Marcar como hechas tareas.	Marcar las tareas ya realizadas para evitar duplicidades.	Alta	Alta
RF-E05	Añadir tareas.	Añadir nuevas tareas para dejar testigo a otros turnos o trabajadores.	Alta	Alta
RF-E06	Reestablecer campos de texto.	Vaciar los campos de texto editable para que la introducción sea más sencilla.	Baja	Baja
RF-E07	Consulta listado de usuarios.	Mostrar la información básica de los usuarios activos.	Alta	Alta
RF-E08	Búsqueda de un usuario concreto.	Mostrar la información completa de un determinado usuario.	Alta	Alta
RF-E09	Añadir medida de constantes	Añadir un nuevo valor de constantes a la base de datos del usuario.	Alta	Alta
RF-E10	Ver gráficas sanitarias.	Simplificar el estudio y análisis sanitario del interno, mostrando los datos de	Media	Media

		forma gráfica.		
RF-E11	Consulta historial médico.	Informar de la historia médica del paciente para su mejor tratamiento.	Alta	Alta
RF-E12	Fotografía del interno.	Mostrar una fotografía del paciente para su rápida identificación.	Baja	Media

Tabla 13: Descripción de los requisitos funcionales que cumple el sistema

2.6.3 Requisitos inversos

ID	Nombre	Descripción
RI-A01	Interfaz de administración.	No existe una interfaz que permita al administrador gestionar la base de datos de manera cómoda y sencilla.
RI-E01	Periodicidad de las tareas.	No se permite añadir tareas con un periodo de repetición.
RI-E02	Notificaciones y alarmas.	No se notifica al usuario cuando una tarea ha caducado o le falta poco para ello.
RI-E03	Login y roles.	No se controla el login en la app para controlar los roles ni las competencias.
RI-E04	Búsqueda por nombre.	No se contempla que varios usuarios activos compartan nombre.
RI-E05	Uso avanzado del NFC.	No se permite el uso de NFC para añadir nuevas tareas ni obtener medidas de dispositivos externos (Telemedicina)

Tabla 14: Descripción de los requisitos inversos del sistema

2.6.4 Requisitos no funcionales

ID	Nombre	Descripción	Prioridad	Necesidad
RNF-E01	Alta en la base de datos.	El administrador puede añadir y editar entradas de la base de datos desde una GUI accesible.	Alta	Alta
RNF-E02	Asignación de tareas	Al añadir una tarea se le asigna a un determinado enfermero.	Media	Media

Tabla 15: Descripción de los requisitos no funcionales que del sistema

2.7 Análisis de seguridad

Este es un proyecto en el que la información almacenada y/o en tránsito, dada su naturaleza, ha de ser tratada de acuerdo a unos estándares de seguridad elevados. Por tanto, al hace uso de un canal inseguro como internet es necesario un estudio detenido de la seguridad de la aplicación y que éste cumpla con los requisitos impuestos por la LOPD (Ley Oficial de Protección de Datos) en materia de datos personales e información sanitaria.

En este apartado se van a analizar y describir los diferentes aspectos del campo de la seguridad que se han tenido en cuenta en cada módulo del proyecto.

Es importante garantizar tanto la confidencialidad de la información como la integridad de los datos. Con el fin de asegurar estos dos aspectos se podrían usar filtros MAC (Media Access Control) además de los mecanismos de login (usuario / contraseña).

Para dotar a la aplicación de disponibilidad y confidencialidad se ha evitado en medida de lo posible el almacenamiento de datos en el propio terminal (a excepción de los datos de la residencia al no tratarse de datos susceptibles). Es por esto que se usa un servidor para esta función al que se accede en remoto.

Por otro lado, se necesita autenticación para garantizar la identidad de los actores intervinientes en la comunicación y evitar la suplantación de la misma. La base de datos se ha diseñado para tal fin como se refleja en el apartado de líneas futuras de trabajo.

Se ha contemplado la posibilidad de emplear un canal seguro como HTTPS (Hypertext Transfer Protocol Secure) para la comunicación, puesto que proporciona integridad, confidencialidad y autenticación en destino.

3 Diseño detallado e implementación

Es necesario recordar que el sistema GeriApp trabaja en su mayor ámbito con un servidor en remoto, al que consulta toda la información a excepción de los datos de la propia residencia. De este modo, se hace estrictamente necesaria una conexión al servidor desde la aplicación móvil en todo momento.

A continuación se muestran los diagramas de clases de los componentes de la arquitectura descrita en el proyecto. Posteriormente se encuentran los diagramas de secuencia que se corresponden con los casos de uso definidos. Por último, se incluye el diagrama relacional de la base de datos, que representa el modelo que se ha empleado para representar la información del sistema.

El sistema en general sigue las premisas de la metodología de trabajo expuestas en el Modelo-Vista-Controlador que se explica en la siguiente imagen:

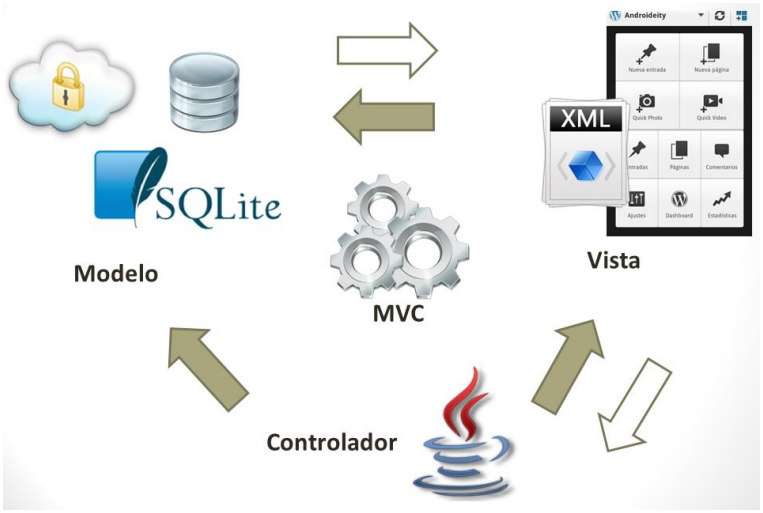


Figura 11: Modelo-Vista-Controlador en Android
http://androideity.com/2012/05/10/la-importancia-del-mvc-en-android

- **Modelo**. Representaciones basadas en la información con la que operara la aplicación. En Java, el modelo viene siendo análogo a los beans reutilizables y permiten hacer las aplicaciones escalables. Se implementan a modo de beans en la aplicación Android y a modo de objetos JSON en la parte del servidor.
- **Vista**. Interfaz con la que interactua el usuario. En Android, las interfaces se construyen en XML.
- **Controlador**. Clases que permiten desplegar y consumir información de y para el usuario. Se programan en lenguaje Java y son el núcleo de la aplicación.

3.1 Diseño de componentes de la aplicación móvil

Los diagramas de clases muestran las diferentes clases que componen el modelo del sistema, las relaciones entre ellas y sus funciones y atributos. En los diagramas se mostrará sólo lo más significativo de cada clase con el fin de exponer la funcionalidad lo más concisamente.

La mayoría de las clases del módulo que se encarga de la interfaz de la aplicación Android extienden de Activity. Cada una de ellas tiene un fichero XML asociado en el que se define el aspecto físico de la interfaz.

Algunos métodos como los relacionados con el ciclo de vida de la aplicación y los relacionados con configuraciones internas se han omitido para evitar redundancia y hacer la explicación lo más clara posible. Estos son:

- **onCreate().** Se llama al crearse el Activity. Es utilizado para inicializar los principales componentes de la misma, indicar la interfaz de usuario a usar y añadir los elementos a la lista de escuchadores de eventos necesaria.
- **onCreateOptionsMenu (Menu menu)** y **onOptionsItemSelected (MenuItem item).** Gestionan los elementos que deben aparecer en la barra de estado de la aplicación y las acciones que se deben realizar cuando uno de ellos es seleccionado.

3.1.1 Diseño del componente "Modelo"

Se trata de un paquete que contiene la totalidad de clases Java del modelo estructural de objetos auxiliares usados en la aplicación. Son un total de 17 clases contenedoras que modelan cada uno de los elementos de las tablas de la base de datos y son meras estructuras con los atributos correspondientes a las columnas de las tablas y sus correspondientes métodos **set(…)** y **get()** para controlarlos.

Además, se ha sobrescrito el método **toString()** para poder imprimir por log o consola los valores almacenados en el objeto.

Figura 12: UML de las clases del paquete modelo

3.1.2 Diseño del componente "ConexionTomcat"

Dentro del paquete conexionTomcat se encuentran las 11 clases diferentes que se encargan de gestionar la conexión con el servlet y obtener la información del JSON recibido. En este apartado se tomará como ejemplo el funcionamiento de la clase ConexionTomcatUsuarios.java, puesto que en el resto de clases del paquete funcionan de forma análoga.

Al tratarse de una operación de **I/O** de datos, las recomendaciones del SDK de Android aconsejan tratarla como una tarea asíncrona y por esto dichas clases extienden de AsyncTask.

```
public class ConexionTomcatUsuarios extends AsyncTask<URL, Void, List<ModeloUsuario>>
```

De este modo, las peticiones del tipo **POST** a la **URL** del servlet se tratan en segundo plano, sin bloquear la aplicación.

El método implementado de la clase AsyncTask es:

protected List<ModeloUsuario> doInBackground(URL... urls). Realiza todo el trabajo en segundo plano. Recibe como argumento la URL para la conexión Apache y devuelve una lista con objetos del tipo indicado en el modelo para cada tipo

de consulta. La lectura del archivo JSON recibido se realiza mediante la función **leerFlujoJson(...).**

El funcionamiento de este conjunto de procedimientos es el siguiente:

Sobre un objeto del tipo ConexionTomcat se realiza una llamada combinada a los métodos **execute()** y **get()** para obtener como devolución una lista de objetos de la clase Modelo requerida.

Se desencadena una cascada de llamada a funciones auxiliares que se encargan de leer e interpretar el flujo de datos de entradas y a partir de éste, construir los objetos.

Figura 13: UML de las clases del paquete conexionTomcat

3.1.3 Diseño del componente "Interfaz"

Bajo este nombre se agrupan las 12 clases que modelan cada una de las 12 interfaces diferentes que posee la aplicación. Todas ellas extienden de Activity e implementan OnClickListener.

Figura 14: UML de las clases del paquete interfaz

En el siguiente diagrama de clases se observa cómo trabaja la aplicación móvil al recibir los datos del servidor.

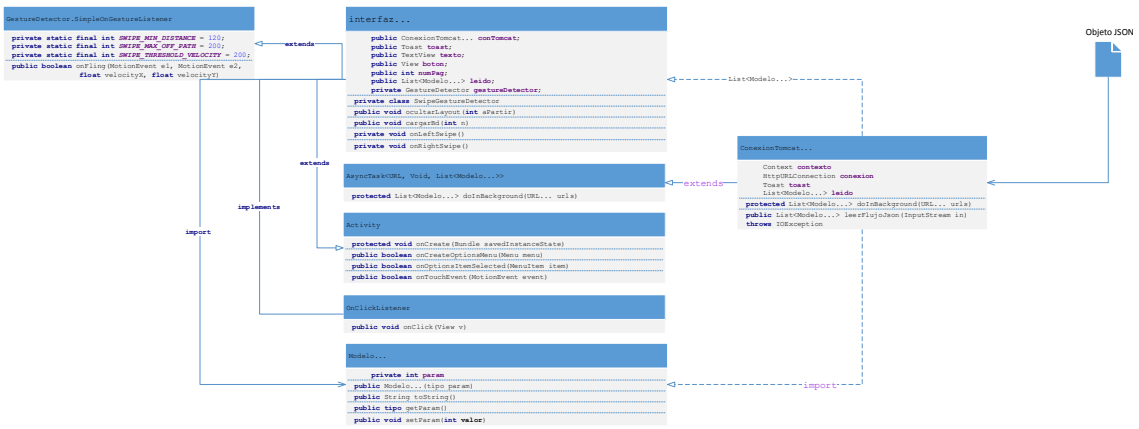


Figura 15: Diagrama de clases del cliente

3.2 Diseño de componentes del servidor

En la parte del servidor se han implementado diferentes servlets que escuchan y atienden peticiones del servicio web.

El control del *pool* de conexiones se ha delegado al servicio Apache-Tomcat.

La función **init(ServletConfig config)** se encarga de tomar los valores del archivo de contexto para la conexión del servidor Apache con la base de datos y de establecer dicho nexo.

Las funciones **doPost(HttpServletRequest request, HttpServletResponse response)** o **doGet(HttpServletRequest request, HttpServletResponse response)** se encargan de:

- Obtener los parámetros recibidos en la URL.
- Crear y ejecutar la sentencia SQL sobre la conexión creada anteriormente.
- Rellenar un objeto JSON con la respuesta obtenida del servidor MySQL.

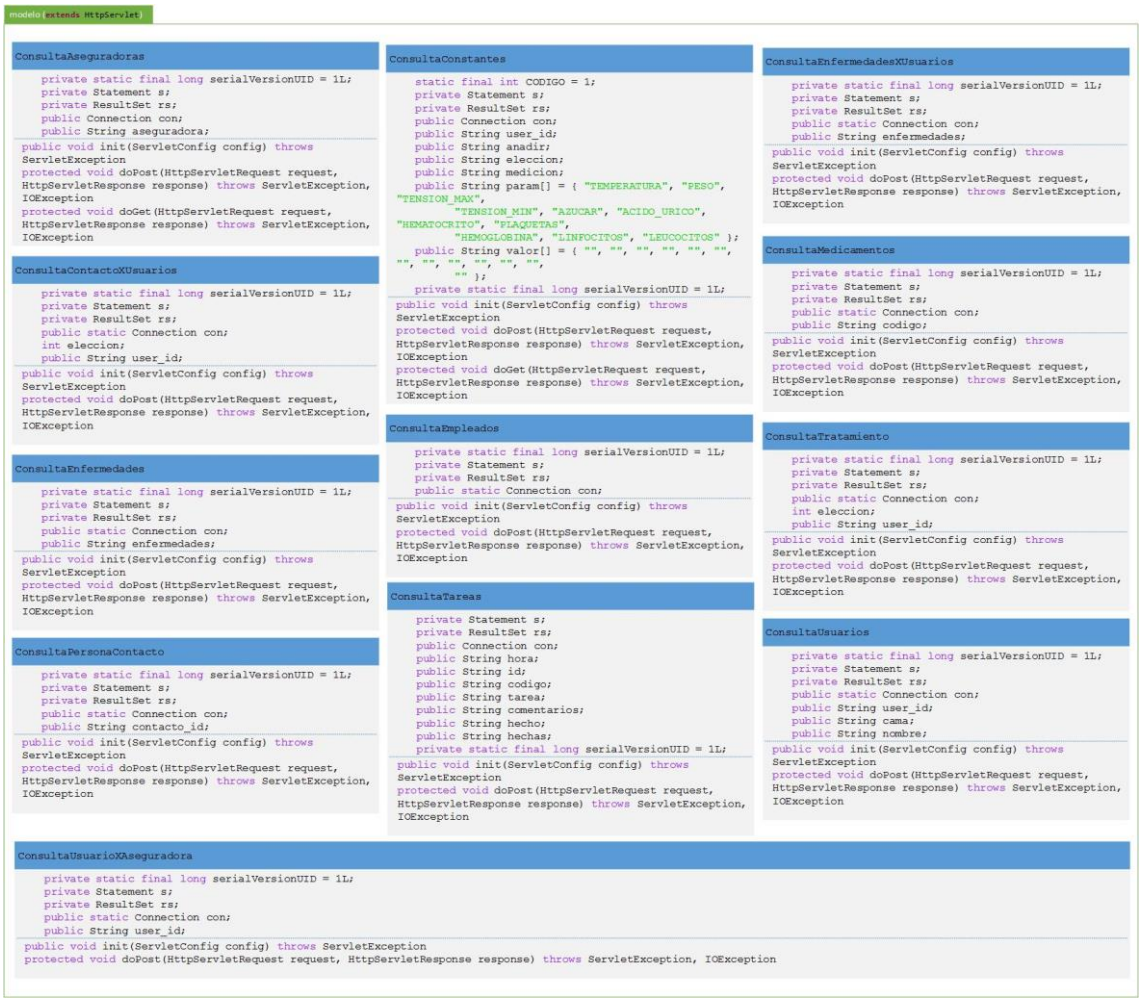


Figura 16: UML de las clases del servidor

En el siguiente diagrama de clases se explica como el servidor obtiene la información de la base de datos y genera el objeto JSON.

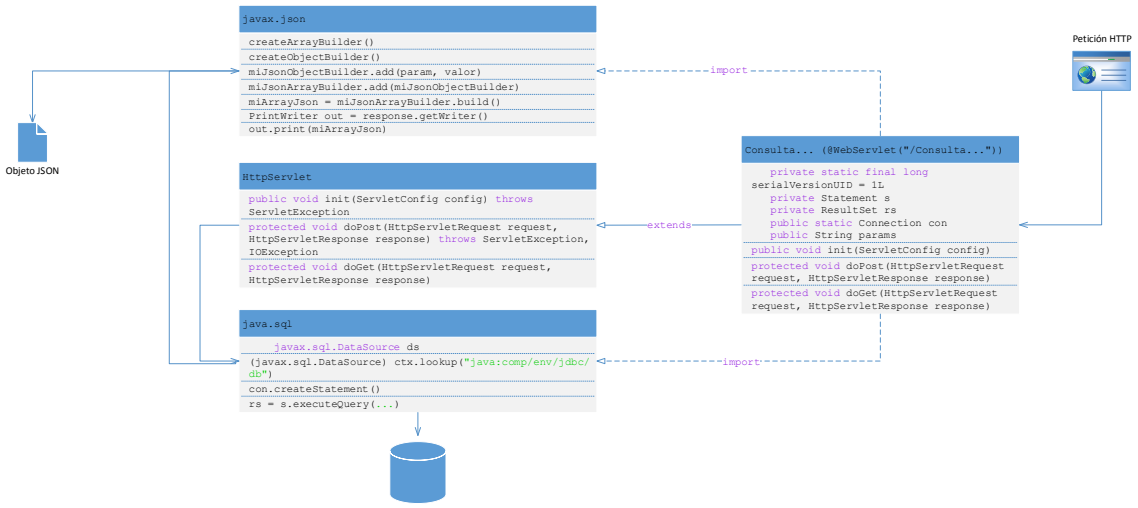


Figura 17: Diagrama de clases del servidor

3.3 Diseño de componentes de la base de datos

En cuanto a la base de datos no ha sido precisa la codificación manual de casi ningún elemento, puesto que la propia herramienta de Workbench proporciona facilidades para crear las sentencias SQL necesarias a partir del esquema relacional indicado.

Se han necesitado un total de 17 tablas:

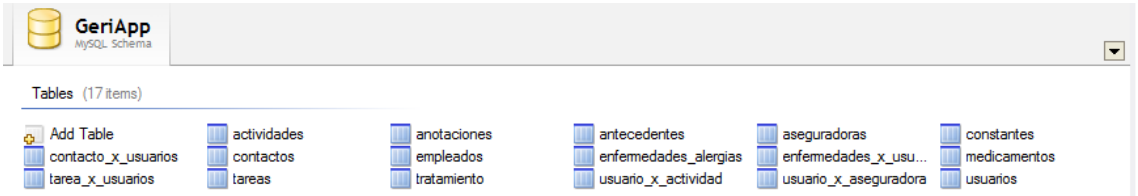


Figura 18: Vista de las tablas que se almacenan en la base de datos del sistema

A continuación, se describen cada una de ellas a excepción de las puramente relacionales o las que se han implementado para trabajos futuros.

Tabla USUARIOS:

Dicha tabla sirve para modelar los internos del centro, empleando como clave primaria un identificador único (ID).

En ella se almacena la información del paciente en variables tipo VARCHAR para las cadenas de texto, tipo DATE para las fechas y BOOLEAN para el campo que se encarga de controlar si el usuario está actualmente interno.

Tabla TRATAMIENTO:

La tabla tratamiento sirve para modelar la ingesta de medicamentos prevista para cada uno de los ancianos del centro, empleando como clave primaria una terna formada por: un identificador único de usuario (ID), un número identificativo del medicamento (MEDICAMENTOS_CODIGO) y una fecha de inicio (FECHA_INICIO).

El resto de campos son enteros que representan el número de tomas diarias que se requieren, los días que dura el tratamiento y la dosis de cada toma.

Tabla TAREAS:

Dicha tabla tiene como fin almacenar la información sobre las tareas propias del centro geriátrico, empleando como clave primaria un identificador entero único (ID_TAREA).

Cada tarea tiene asignado un usuario concreto, una hora límite, dos campos de texto que explican en qué consiste (uno interno y otro externo) y un boolean que sirve para comprobar su estado.

Tabla MEDICAMENTOS:

La tabla medicamentos sirve para modelar las diferentes medicinas que se suelen usar en este tipo de centros. Cada entrada se identifica unívocamente por su clave primaria (CODIGO).

El resto la información de las medicinas se guarda en variables tipo VARCHAR, se trata de campos de texto que explican su nombre, descripción y URL del Vademécum.

Tabla ENFERMEDADES/ALERGIAS:

Dicha tabla contiene información de las diferentes enfermedades más habituales en ancianos. Cada una de ellas se puede diferenciar empleando como clave primaria un identificador único (CODIGO).

Cada enfermedad queda definida por su nombre y un boolean que indica si es permanente o no.

Tabla CONTACTOS:

La tabla contactos sirve para almacenar la información de las personas de contacto de los internos del centro, empleando como clave primaria un identificador único (ID).

El resto de información de estas personas son campos de texto en forma de VARCHAR y un entero que representa el teléfono.

Tabla CONSTANTES:

La siguiente tabla modela las constantes vitales de los internos del centro, empleando como clave primaria una terna formada por un identificador único (ID), un campo DATETIME (FECHA_LECTURA) y un entero que representa el código del usuario que ha tomado la medida.

El resto de campos son enteros que guardan los valores de las constantes medidas.

Tabla ASEGURADORAS:

La tabla *aseguradoras* sirve para almacenar la información de las empresas de seguros de la salud que los usuarios pueden tener contratadas. Se identifican por un entero (ID).

Cada entrada contiene la información propia de la empresa, como su nombre, dirección y teléfono de contacto. En la figura 19 se puede observar el Modelo Entidad (MER) de la base de datos.

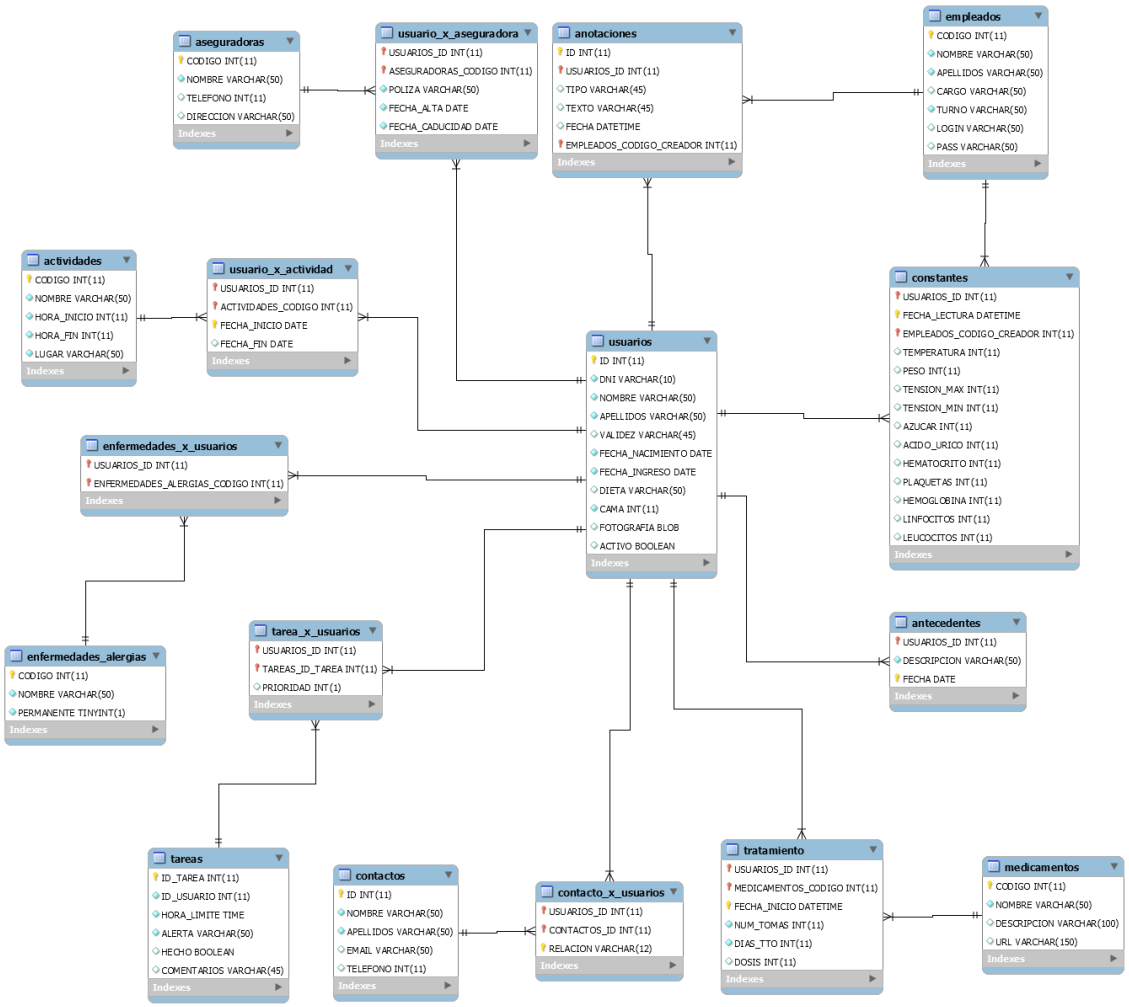


Figura 19: Modelo Entidad (MER)

3.4 Diagramas de secuencia

Esta sección recoge los diagramas de secuencia, que modelan el flujo lógico del sistema de manera ordenada y visual. De este modo cada diagrama se corresponde con un caso de uso de los especificados anteriormente.

3.4.1 Envío y recepción de información entre cliente y servidor

Cada clase contenida dentro del paquete interfaz tiene como atributo un objeto de la clase análoga del paquete conexionTomcat, que al extender de AsyncTask permite invocar al método **execute(Params…)** para realizar la tarea de forma asíncrona.

En el paquete conexionTomcat y desencadenado por la llamada al método **execute(…)**, se ejecuta el método **doInBackground(Params…)** que es el encargado de enviar la petición al servidor y obtener la respuesta.

La petición llega al servidor al método **doPost()** de la clase adecuada (aquella que controla el servlet para cada tipo de datos definidos en el modelo) Es en ese momento cuando se prepara la consulta a la base de datos y se ejecuta la misma. Con la respuesta se construye una estructura JSON, que es enviada de la aplicación móvil al objeto de la clase conexionTomcat, que lo trata para convertirlo en una lista de objetos de la clase necesaria del paquete modelo.

De este modo, la repuesta final es el valor devuelto por **execute(…).get()** de la clase del paquete interfaz correspondiente.

De forma esquematica quedaría:

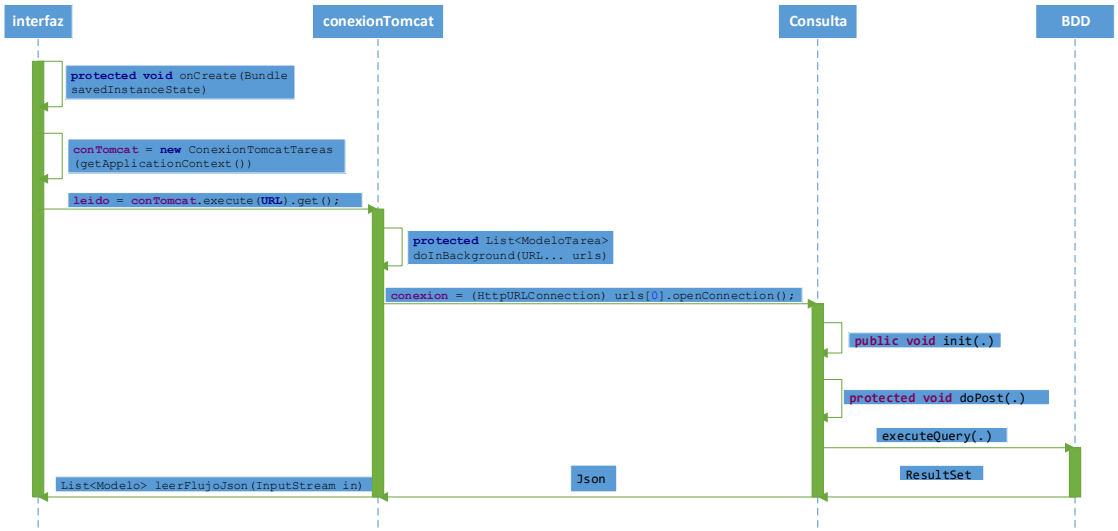


Figura 20: Diagrama UML del envío y recepción de información entre cliente y servidor

CU-E01: Consultar lista de tareas pendientes/hechas

Para poder ver el listado de tareas pendientes el agente usuario debe seleccionar en el menú principal el icono dedicado al tal fin. Esto hace que la aplicación cree un nuevo Activity para mostrar dicha información bajo las indicaciones de la clase Java TareasPendientes.

La llamada a **startActivity(.)** hace que se ejecute el código en del método **onCreate(.)** de la clase TareasPendientes, de esta forma se inicializan las variables y se crea la URL para la petición HTTP hacia el servlet.

Se almacena la respuesta del servidor en un Array List, que es recorrido por la función **cargarBd(int n)**, para ir ubicando la información en la pantalla para el usuario.

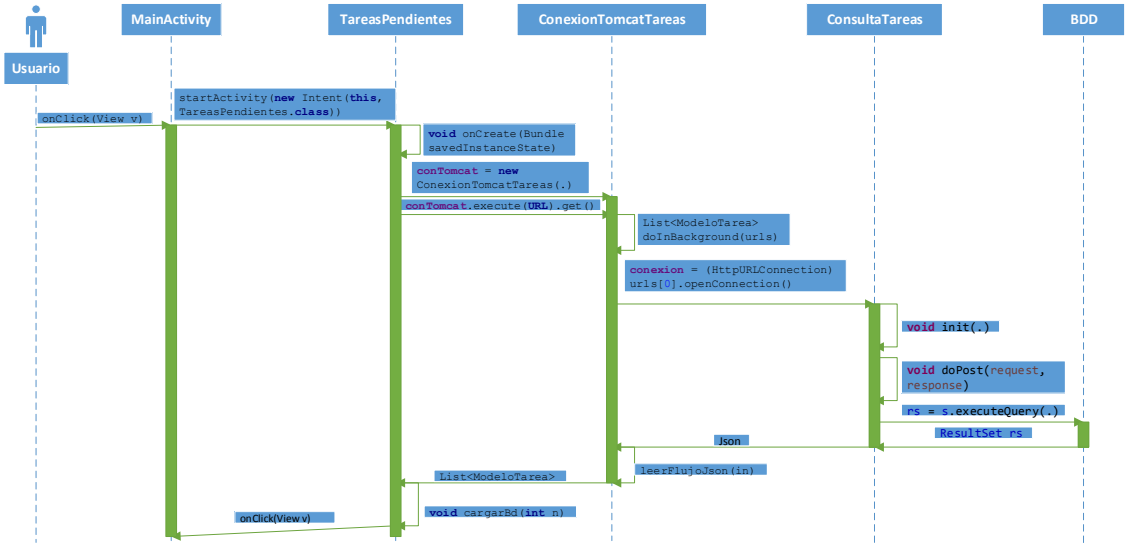


Figura 21: Diagrama UML del CU-E01

Al obtenerse la totalidad de las entradas de la tabla *tareas* (las que tienen el flag hecho a 0 o a 1 dependiendo del caso) en la carga del **Activity**, se puede usar los botones con forma de flechas y la acción de swipe para mostrar los siguientes o anteriores elementos de la lista.

CU-E02: Añadir tarea para hacer

En este caso de uso el usuario debe seleccionar el botón con forma de símbolo de suma para acceder al menú que le permite añadir una nueva tarea a la base de datos. Se abre un nuevo **Acivity (tareas_pendientes.xml)** con un reloj seleccionable mediante scroll y tres campos de texto editables.

Tras rellenar la información requerida de forma correcta el usuario elige la opción de guardar y se procede a almacenar la información en la base de datos.

Los valores obtenidos de la interfaz se codifican en la URL para que sean leídos por el servlet.

El esquema es similar al anterior.

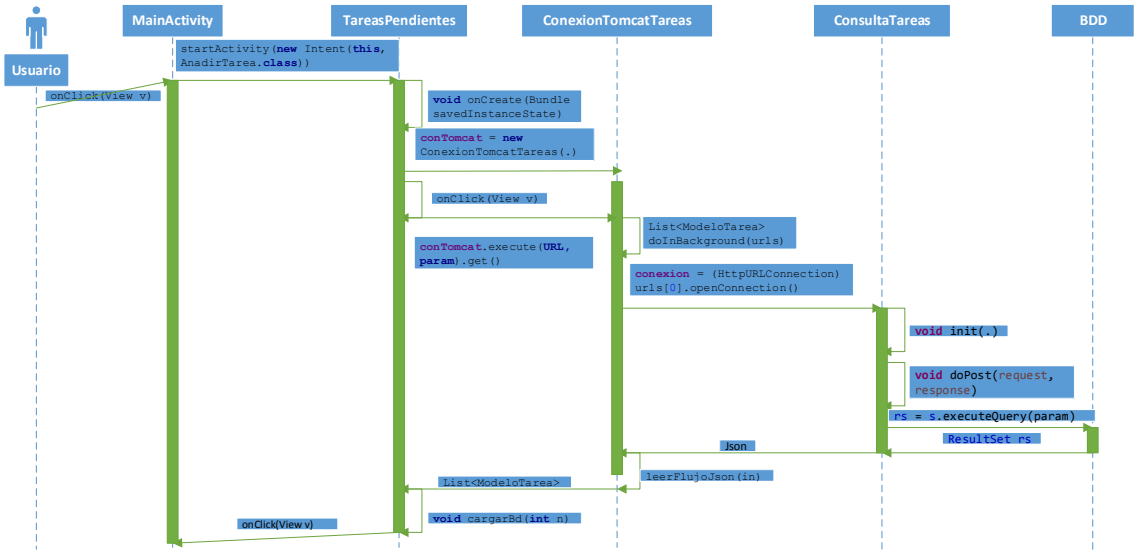


Figura 22: Diagrama UML del CU-E02

CU-E03: Consultar lista de usuarios registrados

Este caso, al ser totalmente idéntico al CU-E01 en su funcionamiento, no va a ser detallado exhaustivamente ni descrito con diagramas, puesto que se entiende al ver que simplemente se cambia el rol de las tareas por usuarios.

CU-E04: Buscar un usuario determinado

Este caso de uso es peculiar, puesto que se da en tres hilos diferentes de la ejecución. El agente usuario puede llegar a él desde la interfaz principal, ya sea realizando una búsqueda por cama o por NFC, o desde la opción de búsqueda en la vista de los usuarios registrados, bien sea buscando por nombre o por número de cama.

De cualquiera de las dos formas, el funcionamiento es casi idéntico:

- Se genera una petición para obtener el ID del interno a partir del criterio de búsqueda.
- Se pasa el ID al Activity de FichaPersonal como parámetro.
- El nuevo Activity realiza otra petición para obtener los datos del usuario recibido.
- Se representan los datos obtenidos en pantalla.

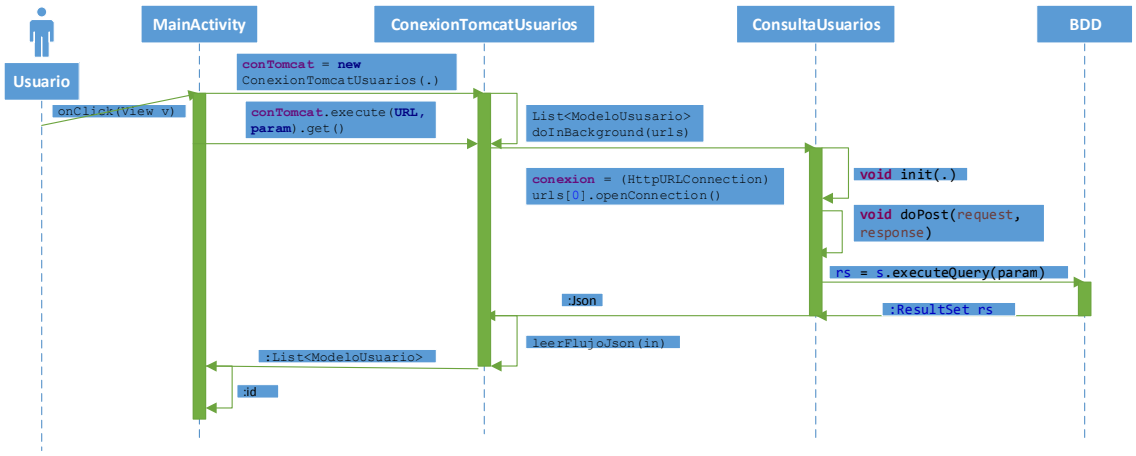


Figura 23: Diagrama UML del CU-E04 para la petición del ID

Una vez con el ID se realiza una nueva búsqueda:

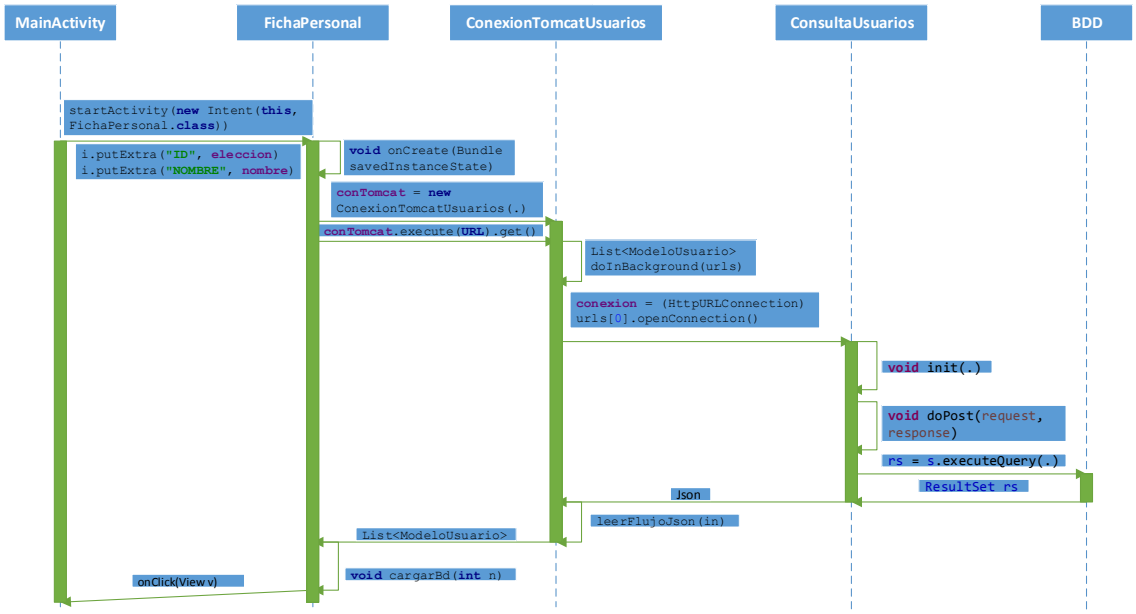


Figura 24: Diagrama UML del CU-E04 para la obtención del resto de información a partir del ID

CU-E05: Contactar con la persona asignada

Tras consultar la ficha personal de cualquier interno, las enfermeras pueden contactar con la persona de contacto del mismo mediante una llamada telefónica o mediante un correo electrónico.

Como la parte de la obtención de los datos de contacto es muy similar, sólo se van a presentar los diagramas de la gestión de la comunicación.

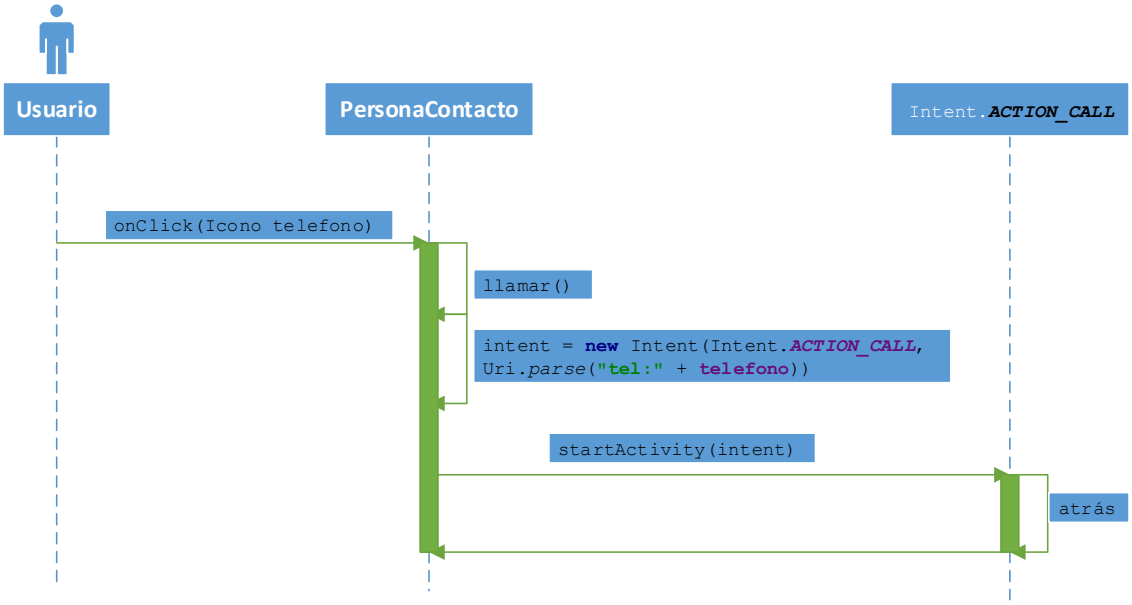


Figura 25: Diagrama UML del CU-E05 para la llamada

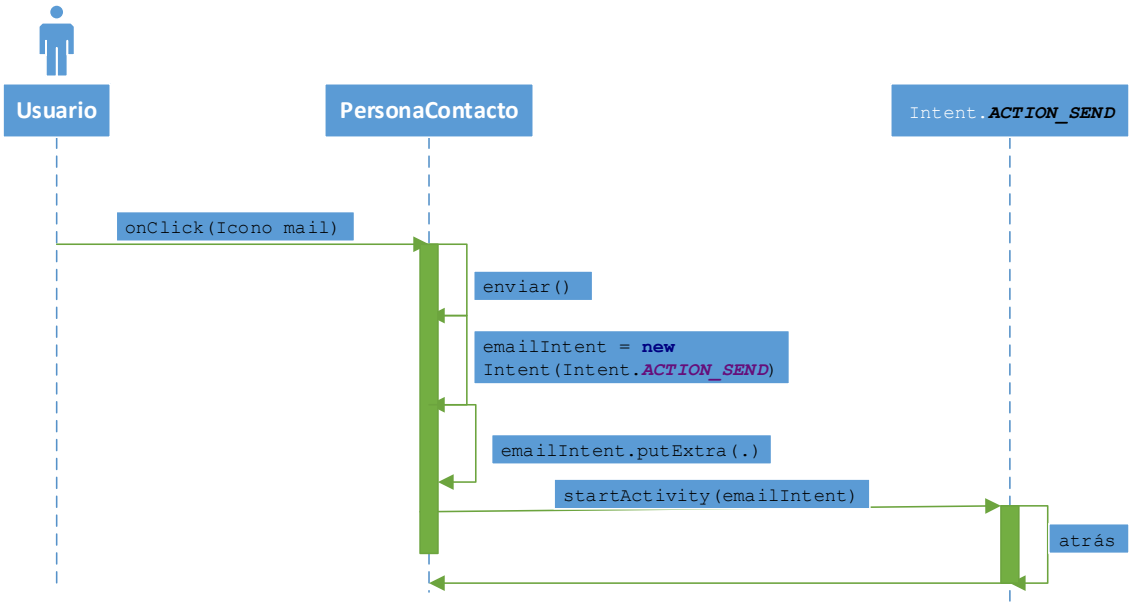


Figura 26: Diagrama UML del CU-E05 para el envío de un correo electrónico

CU-E06: Añadir medidas de constantes

Una vez que el usuario se encuentra en la vista de la información médica del interno del geriátrico, debe seleccionar la opción de añadir para que se le ofrezca la nueva interfaz con los textos editables que ha de rellenar.

Inicialmente se precarga con los últimos valores para que sólo se modifiquen los que han variado.

Al seleccionar uno de los "edit text", éste cambia de color para hacer visible que es nuevo y una vez acabada la edición se elige la opción de guardar.

En el método **anadirBd()** de la clase NuevaMedida se comprueba previamente que los valores sean correctos y se construye una URL con todos ellos como parámetro. La forma de enviar la petición y tratar la respuesta es la misma que a la hora de añadir tareas.

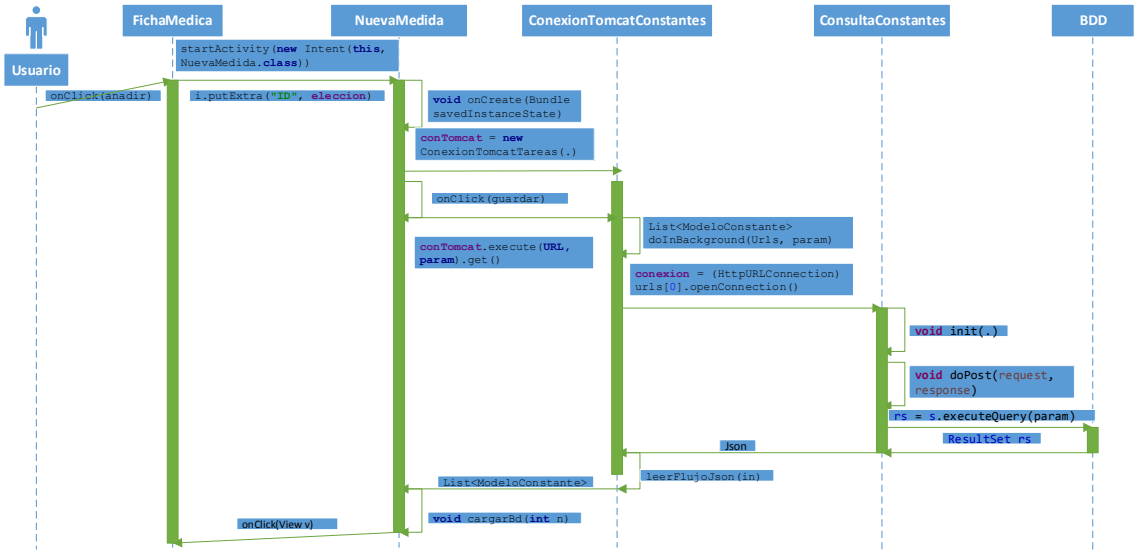


Figura 27: Diagrama UML del CU-E06

4 Implantación y pruebas del software

En este capítulo se describe el proceso de desarrollo del software a partir del diseño detallado del sistema y de las necesidades impuestas por la arquitectura.

4.1 Codificación

Este apartado describe los aspectos más importantes del proceso de creación del código en los aspectos más importantes y relevantes del sistema llevado a cabo en este proyecto.

4.1.1 Envío de datos de la aplicación al servidor

Como el envío de datos desde GeriApp un proceso muy importante a la vez que repetitivo se va a exponer el código referente al envío de una petición como ejemplo, pudiéndose extrapolar el procedimiento al resto casi en su totalidad.

Se toma como ejemplo el envío de datos propio de a consulta del listado de usuarios registrados:

```
public class BaseDeDatos extends Activity implements OnClickListener {
        public ConexionTomcatUsuarios conTomcat;
        public List<ModeloUsuario> leido;
        …
protected void onCreate(Bundle savedInstanceState) {
        ConnectivityManager connMgr = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            conTomcat = new ConexionTomcatUsuarios(getApplicationContext());
            leido = conTomcat.execute(new URL("http://" + MainActivity.IP +
        ":8080/GeriApp/ConsultaUsuarios")).get();
        cargarBd(numPag);
        }
        …
}
public void cargarBd(int n) {
         ModeloUsuario mi usuario;
         Iterator<ModeloUsuario> it = leido.iterator();

           while (it.hasNext()) {
               mi_usuario = it.next();
           …
           }
…
}
```

En la clase BaseDeDatos se tienen dos atributos de clase, uno para la conexión y otro para la devolución. En ella, en concreto en el método **onCreate(.)** se inicializan con los valores necesarios para el servlet. Una vez obtenida la respuesta se trata el ArrayList en el método **cargarBd(.)**, imprimiendo por pantalla la información obtenida.

El objeto de la clase conexionTomcatUsuarios comienza su ejecución asíncrona con la llamada a su método **execute(.)** y se encarga de tratar los datos serializados en el objeto JSON enviado por el servidor.

```
public class ConexionTomcatUsuarios extends AsyncTask<URL, Void, List<ModeloUsuario>>
{
HttpURLConnection conexion;
…
protected List<ModeloUsuario> doInBackground(URL... urls) {
    List<ModeloUsuario> lectura = new ArrayList<>();
    conexion = (HttpURLConnection) urls[0].openConnection();
    InputStream in = new BufferedInputStream(conexion.getInputStream());
    lectura = leerFlujoJson(in);
    leido = new ArrayList<ModeloUsuario>(lectura);
…
}
public List<ModeloUsuario> leerFlujoJson(InputStream in) throws IOException {
    android.util.JsonReader reader = new android.util.JsonReader(new
InputStreamReader(in, "UTF-8"));
    try {
        return leerArray(reader);
    } finally {
        reader.close();
    }
}
…
```

Ambas clases hacen uso de las clases del paquete modelo que encapsulan cada elemento necesario de las tablas de la base datos en forma de objetos, Java Beans, con atributos correspondientes a campos.

4.1.1 Envío de datos del servidor a la aplicación

El servidor consta de un servidor web Apache-Tomcat escuchando en el puerto 8080 en el que se encuentran alojados los servlets correspondientes a cada tipo de petición.

El servidor web escucha las peticiones en forma de URL y las traduce para ver que servlet es el que se requiere para cada caso.

Una vez hecho esto se toman los parámetros pasados junto con la dirección web y con ellos se construye la petición SQL. Dicha sentencia se ejecuta y con el resultado obtenido se construye el objeto JSON a entregar al cliente.

El código se detalla en la sección posterior encargada de exponer el funcionamiento de los servlets.

4.1.2 Otras secciones de código interesante

Es preciso detenerse a explicar ciertas funciones y métodos implementados, ya sea por su complejidad o por su funcionalidad.

USO DE SWIPE EN PANTALLA

Para avanzar en las lista o retroceder se ha implementado la función de *swipe* lateral:

```
gestureDetector = new GestureDetector(new SwipeGestureDetector());
private class SwipeGestureDetector extends GestureDetector.SimpleOnGestureListener {
if (diff > SWIPE_MIN_DISTANCE
        && Math.abs(velocityX) > SWIPE_THRESHOLD_VELOCITY) {
    onLeftSwipe();
} else if (-diff > SWIPE_MIN_DISTANCE
        && Math.abs(velocityX) > SWIPE_THRESHOLD_VELOCITY) {
    onRightSwipe();
}
```

USO DE NOTIFICACIONES

Se han usado los objetos **Toast** de Android para informar al usuario de los resultados de las operaciones en forma de notificaciones emergentes.

```
toast = Toast.makeText(getApplicationContext(), "ERROR AL MOSTRAR INFORMACIÓN
MÉDICA", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.CENTER, 0, 0);
toast.show();
```

OBTENCION DE IMÁGENES DEL SERVIDOR

En la carpeta *imgs* del servidor Apache-Tomcat se encuentran alojadas las imágenes de los usuarios internos. Se accede a ellas de forma dinámica, puesto que el nombre de los archivos sigue un patrón conocido.

```
imageUrl = new URL(direccion);
conn = (HttpURLConnection) imageUrl.openConnection();
conn.connect();
BitmapFactory.Options options = new BitmapFactory.Options();
options.inSampleSize = 0;
Bitmap imagen = BitmapFactory.decodeStream(conn.getInputStream(), new Rect(0, 0, 0,
0), options);
foto.setImageBitmap(imagen);
```

USO DE LA LIBRERÍA PARA GRAFICAR

Para poder representar en forma de gráficas los valores de las constantes de los usuarios internos en el geriátrico se ha optado por usar una librería jar gratuita, sencilla y compatible: AndroidPlot (http://androidplot.com).

Se cargan los valores de la base de datos y posteriormente se crean los arrays que se pasan a la función encargada del plot en pantalla. Además se marcan los valores que definen los rangos normales de cada parámetro para su mejor entendimiento.

```
import com.androidplot;
public void pintarGráfica(String que) {
    grafica = (XYPlot) findViewById(R.id.grafica);
    grafica.setTitle(que);
if (que == "TEMPERATURA") {
    series1Numbers = cargarConst(1);
    series2Numbers = llenarConstante(35, series1Numbers.length);
XYSeries series1 = new SimpleXYSeries(
        Arrays.asList(series1Numbers),
        SimpleXYSeries.ArrayFormat.Y_VALS_ONLY,
        que);

LineAndPointFormatter series1Format = new LineAndPointFormatter(
        Color.RED,                              // Color de la línea
        Color.BLACK,                            // Color de los puntos
        null,                                   // Color de relleno
        new PointLabelFormatter(Color.WHITE));  // Color del texto
grafica.addSeries(series1, series1Format);
```

USO DE SHARED PREFFERENCES

Tal y como se ha explicado anteriormente, se usa este método para almacenar la información de la residencia y poder cargarla de forma dinámica a la hora de mandar un correo electrónico. Además se puede usar para almacenar valores constantes y así no tener que modificar el código en función de estos.

```
preferenceSettings = getSharedPreferences(ARCHIVO, PREFERENCE_MODE_PRIVATE);

preferenceEditor = preferenceSettings.edit();
preferenceEditor.putString("NOMBRE", "__Nombre__Residencia__");
preferenceEditor.putString("DIRECCION", "__Dirección__Residencia__");
preferenceEditor.putString("TELEFONO", "__Teléfono__Residencia__ ");
preferenceEditor.commit();
```

```
residenciaNombre = preferenceSettings.getString("NOMBRE", "Sin nombre");
residenciaDireccion = preferenceSettings.getString("DIRECCION", "Sin dirección");
residenciaTelefono = preferenceSettings.getString("TELEFONO", "Sin teléfono");
```

USO DE LA LLAMADA

Para poder contactar con el familiar o persona de contacto predefinida se ha implementado la opción de llamada directa (con los consiguientes permisos).

```
private void llamar() {
        TelephonyManager manager = (TelephonyManager)
getApplicationContext().getSystemService(Context.TELEPHONY_SERVICE);
        if (manager.getPhoneType() != TelephonyManager.PHONE_TYPE_NONE) {
            Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:" +
telefono));
            startActivity(intent);
```

FUNCIÓN DE ENVÍO DE EMAIL

La otra posibilidad de comunicación es la de enviar un email con un cuerpo predefinido a la persona de contacto.

```
private void enviar(String[] to, String[] cc, String asunto, String mensaje) {
    Intent emailIntent = new Intent(Intent.ACTION_SEND);
    emailIntent.setData(Uri.parse("mailto:"));
    emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
    emailIntent.putExtra(Intent.EXTRA_CC, cc);
    emailIntent.putExtra(Intent.EXTRA_SUBJECT, asunto);
    emailIntent.putExtra(Intent.EXTRA_TEXT, mensaje);
    emailIntent.setType("message/rfc822");
    startActivity(Intent.createChooser(emailIntent, "Email "));
}
```

USO DEL NFC

Como funcionalidad extra y como base a futuras funcionalidades se ha decidido implementar el uso de NFC para leer etiquetas con ID y realizar tareas de búsqueda de forma más rápida.

Para ello se ha usado una clase auxiliar que se encarga de la lectura y decodificación del NFC-tag.

```
import android.nfc
…
public static void setupForegroundDispatch(final Activity activity, NfcAdapter
adapter) {
    if (adapter != null) {
        final Intent intent = new Intent(activity.getApplicationContext(),
activity.getClass());
        intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP);

        final PendingIntent pendingIntent =
PendingIntent.getActivity(activity.getApplicationContext(), 0, intent, 0);

        IntentFilter[] filters = new IntentFilter[1];
        String[][] techList = new String[][]{};

        filters[0] = new IntentFilter();
        filters[0].addAction(NfcAdapter.ACTION_NDEF_DISCOVERED);
        filters[0].addCategory(Intent.CATEGORY_DEFAULT);
        try {
            filters[0].addDataType(MIME_TEXT_PLAIN);
        } catch (MalformedMimeTypeException e) {
            throw new RuntimeException("COMPRUEBE EL TIPO MIME");
        }

        adapter.enableForegroundDispatch(activity, pendingIntent, filters,
techList);
    }
}
```

```
private class NdefReaderTask extends AsyncTask<Tag, Void, String> {

    @Override
    protected String doInBackground(Tag... params) {
        Tag tag = params[0];

        Ndef ndef = Ndef.get(tag);
        if (ndef == null) {
            return null;
        }

        NdefMessage ndefMessage = ndef.getCachedNdefMessage();

        NdefRecord[] records = ndefMessage.getRecords();
        for (NdefRecord ndefRecord : records) {
            if (ndefRecord.getTnf() == NdefRecord.TNF_WELL_KNOWN &&
Arrays.equals(ndefRecord.getType(), NdefRecord.RTD_TEXT)) {
                try {
                    return readText(ndefRecord);
                } catch (UnsupportedEncodingException e) {
                    toast = Toast.makeText(getApplicationContext(), "CODIFICACIÓN NO
SOPORTADA", Toast.LENGTH_SHORT);
                    toast.setGravity(Gravity.CENTER, 0, 0);
                    toast.show();
                }
            }
        }
        return null;
    }

    private String readText(NdefRecord record) throws UnsupportedEncodingException {

        byte[] payload = record.getPayload();

        String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16";

        int languageCodeLength = payload[0] & 0063;

        return new String(payload, languageCodeLength + 1, payload.length -
languageCodeLength - 1, textEncoding);
    }

    protected void onPostExecute(String result) {
        if (result != null) {
            leidoNFC = Integer.parseInt(result);
            System.out.println("LEIDO " + leidoNFC);
        }
    }
}
```

USO DE SERVLETS

En la parte del servidor se han usado servlets para atender a las peticiones. A continuación se expone su código genérico:

```
@WebServlet("/ConsultaUsuarios")
public class ConsultaUsuarios extends HttpServlet {

public void init(ServletConfig config) throws ServletException {
      super.init();
      javax.naming.InitialContext ctx = null;
      javax.sql.DataSource ds = null;
         ctx = new javax.naming.InitialContext();
         ds = (javax.sql.DataSource) ctx.lookup("java:comp/env/jdbc/db");
         con = ds.getConnection();
      …

}

protected void doPost(HttpServletRequest request,
         HttpServletResponse response) throws ServletException, IOException {
      response.setContentType("application/json");
      response.setHeader("Cache-Control", "nocache");
      response.setCharacterEncoding("utf-8");

         nombre = request.getParameter("NOMBRE");
         s = con.createStatement();

         if (nombre != null) {
            rs = s.executeQuery("SELECT * FROM usuarios WHERE NOMBRE='"
                  + nombre + "'");
         }
          …
         }
         else {
            rs = s.executeQuery("SELECT * FROM usuarios");
         }

         JsonArray miArrayJson;
         JsonArrayBuilder miJsonArrayBuilder = Json.createArrayBuilder();
         JsonObjectBuilder miJsonObjectBuilder = Json.createObjectBuilder();

         while (rs.next()) {
            miJsonObjectBuilder
                  .add("id", rs.getInt("ID"))
                  …

            miJsonArrayBuilder.add(miJsonObjectBuilder);
         }

         miArrayJson = miJsonArrayBuilder.build();
         PrintWriter out = response.getWriter();
         out.print(miArrayJson);
      }
   }
```

4.2 Banco de pruebas

En este apartado se indican las pruebas de aceptación llevadas a cabo para determinar si cumple con los requisitos especificados y a la vez verificar el correcto funcionamiento del sistema completo. La totalidad de las pruebas ha sido testeada con dispositivos físicos y virtuales, además se ha comprobado el correcto funcionamiento tanto con IP de local host o desarrollo (**10.0.2.2**) como en remoto con la IP del servidor.

4.2.1 Formato de las pruebas

A continuación se indica el formato de las pruebas de aceptación que se seguido para las siguientes secciones.

Identificador	Descripción de la prueba
Objetivos	Propósitos de la prueba.
Requisitos cubiertos	Requisitos que se cumplen tras la realización satisfactoria de la prueba.
Precondiciones	Condiciones necesarias que se deben dar en el sistema o en el dispositivo que se empleará para la prueba.
Secuencia	Pasos necesarios para realizar la prueba.
Resultado esperado	Resultado esperado tras llevar a cabo la presente prueba.

Tabla 16: Modelo para el formato de las pruebas realizadas

4.2.2 Pruebas de aceptación

PRU-E01	Consultar la información de la aplicación
Objetivos	Comprobar que se ejecuta un nuevo Activity y se carga el texto almacenado en el fichero strings.xml.
Requisitos cubiertos	• RF-E01: Consulta de la información de la app.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono ACERCA DE…
Resultado esperado	Se muestra la nueva pantalla con toda la información de la app y de su creador.

Tabla 17: Tabla con la información de la prueba PRU-E01

PRU-E02	Consultar tareas sin hacer
Objetivos	Comprobar que se ejecuta un nuevo Activity y se cargan las tareas pendientes de la base de datos.
Requisitos cubiertos	• RF-E02: Consulta tareas pendientes.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono TAREAS PENDIENTES.
Resultado esperado	Se muestra la lista de tareas pendientes. Se puede avanzar/retroceder en la lista.

Tabla 18: Tabla con la información de la prueba PRU-E02

PRU-E03	Consultar tareas ya hechas
Objetivos	Comprobar que se ejecuta un nuevo Activity y se cargan las tareas ya hechas de la base de datos.
Requisitos cubiertos	• RF-E02: Consulta tareas pendientes. • RF-E03: Consulta tareas hechas.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono TAREAS PENDIENTES. 3. Seleccionar la opción de Ver Hechos en la barra lateral.
Resultado esperado	Se muestra la lista de tareas hechas. Se puede avanzar/retroceder en la lista.

Tabla 19: Tabla con la información de la prueba PRU-E03

PRU-E04	Marcar tarea como hecha
Objetivos	Comprobar que se accede a la base de datos para marcar como hechas las tareas seleccionadas.
Requisitos cubiertos	• RF-E02: Consulta tareas pendientes. • RF-E04: Marcar como hechas tareas.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono TAREAS PENDIENTES. 3. Marcar los checkboxs de las tareas que se quieran dejar como hechas. 4. Seleccionar la opción de Marcar hecho en la barra lateral.
Resultado esperado	Se cambian en la base de datos las entradas seleccionadas y se actualiza la lista sin dichas tareas.

Tabla 20: Tabla con la información de la prueba PRU-E04

PRU-E05	Añadir nueva tarea
Objetivos	Comprobar que se accede a la base de datos para añadir nuevas entradas a la tabla tareas.
Requisitos cubiertos	• RF-E05: Añadir tareas. • RF-E06 Restablecer campos de texto.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono AÑADIR TAREA. 3. Seleccionar el icono Descartar si se quieren vaciar los campos. 4. Introducir los valores que se quieren almacenar. 5. Seleccionar la opción de Guardar en la barra lateral.
Resultado esperado	Tras comprobar que los campos son coherentes se añade en la base de datos una entrada para dicha tarea.

Tabla 21: Tabla con la información de la prueba PRU-E05

PRU-E06	Consultar lista de usuarios registrados
Objetivos	Comprobar que se accede a la base de datos para leer la totalidad de las entradas de la tabla usuarios.
Requisitos cubiertos	• RF-E07: Consulta listado de usuarios.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono BASE DE DATOS.
Resultado esperado	Se muestra la lista de usuarios registrados. Se puede avanzar/retroceder en la lista.

Tabla 22: Tabla con la información de la prueba PRU-E06

PRU-E07	Buscar usuario por número de cama o nombre
Objetivos	Comprobar que se accede a la base de datos para obtener un recurso concreto.
Requisitos cubiertos	• RF-E07: Consulta listado de usuarios. • RF-E08: Búsqueda de un usuario concreto.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono BASE DE DATOS. 3. Seleccionar el icono de la lupa. 4. Rellenar el campo nombre o cama. 5. Pulsar de nuevo en Buscar. 6. Introducir el número de cama en la interfaz

	principal. 7. Leer desde NFC en la interfaz principal. 8. Seleccionar el botón de BUSCAR. 9. Seleccionar de la vista de usuarios registrados.
Resultado esperado	Se abre la vista de FichaPersonal con la información de usuario requerido.

Tabla 23: Tabla con la información de la prueba PRU-E07

PRU-E08	**Consultar información personal y médica**
Objetivos	Comprobar que se accede a la base de datos para obtener un recurso concreto.
Requisitos cubiertos	• RF-E07: Consulta listado de usuarios. • RF-E08: Búsqueda de un usuario concreto. • RF-E11: Consulta historial médico.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Seleccionar el icono BASE DE DATOS. 3. Seleccionar el nombre desde la lista de usuarios registrados. 4. Seleccionar el icono de la lupa. 5. Rellenar el campo nombre o cama. 6. Pulsar de nuevo en Buscar. 7. Introducir el número de cama en la interfaz principal. 8. Leer desde NFC en la interfaz principal. 9. Seleccionar el botón de BUSCAR.
Resultado esperado	Se abre la vista de FichaPersonal con la información de usuario requerido, se puede acceder a los datos médicos seleccionado la opción correspondiente.

Tabla 24: Tabla con la información de la prueba PRU-E08

PRU-E09	**Consultar fotografía**
Objetivos	Comprobar que se accede a las imágenes remotas alojadas en el servidor Apache.
Requisitos cubiertos	• RF-E07: Consulta listado de usuarios. • RF-E08: Búsqueda de un usuario concreto.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Acceder a la ficha personal de un usuario como en la prueba PRU-E08. 3. Seleccionar la opción de foto del menú lateral.
Resultado esperado	Se conecta al servidor y muestra una imagen de ejemplo con el ID del usuario.

Tabla 25: Tabla con la información de la prueba PRU-E09

PRU-E10	Contactar con persona asignada
Objetivos	Comprobar que se pueden ejecutar funciones externas como son llamadas o envío de emails.
Requisitos cubiertos	• RF-E07: Consulta listado de usuarios. • RF-E08: Búsqueda de un usuario concreto.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android y conexión con el servidor.
Secuencia	1. Abrir la aplicación. 2. Acceder a la ficha personal de un usuario como en la prueba PRU-E08. 3. Seleccionar la opción de Contacto del menú lateral. 4. Seleccionar Llamar a o Enviar Email.
Resultado esperado	Se abren correctamente los dos interfaces de comunicación. De llamada si el dispositivo lo permite y el gestor de correo con el contenido pre-relleno.

Tabla 26: Tabla con la información de la prueba PRU-E10

PRU-E11	Consultar historial gráficamente
Objetivos	Comprobar que se acceden a las constantes de la base de datos y se maneja la librería de graficar.
Requisitos cubiertos	• RF-E07: Consulta listado de usuarios. • RF-E08: Búsqueda de un usuario concreto. • RF-E10: Ver gráficas sanitarias. • RF-E11: Consulta historial médico.
Precondiciones	Tener previamente instalada la aplicación en el dispositivo Android, conexión con el servidor e importada correctamente la librería encargada de graficar.
Secuencia	1. Abrir la aplicación. 2. Acceder a la ficha personal de un usuario como en la prueba PRU-E08. 3. Seleccionar la opción de Datos Médicos del menú lateral. 4. Seleccionar la opción de Gráficas del menú lateral. 5. Seleccionar una constante de la lista.
Resultado esperado	Se cargan los datos y se grafican dinámicamente los valores.

Tabla 27: Tabla con la información de la prueba PRU-E11

5 Implicaciones legales del proyecto

En este capítulo se describen las implicaciones legales que supone a los usuarios el uso de la aplicación. Además se presentan las condiciones del servicio y la política de privacidad que aceptan los usuarios al instalar GeriApp.

5.1 Utilización del sistema

Tal y como se explica al inicio del documento, la aplicación desarrollada es del tipo *mHealth (mobile health)*, por lo que maneja datos médicos de los usuarios, tales como su medicación, enfermedades, constantes vitales o seguro médico. Esta información es confidencial y personal, por lo que es necesario aplicar el marco jurídico y legal vigente en la Unión Europea para dicha información. En este caso concretamente la Directiva 95/46/CE del Parlamento Europeo y del Consejo, de 24 de Octubre de 1995.

Actualmente la Unión Europea está debatiendo el tratamiento legal y resto de consideraciones jurídicas que se deben aplicar al ámbito de las aplicaciones sanitarias en general.

Así pues, el tratamiento de la información de carácter personal de la aplicación GeriApp (tanto en la tableta y en el servidor como al tránsito) se ajusta a los principios y obligaciones establecidos en la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal (LOPD). Además se cumple con su Reglamento de desarrollo, aprobado por Real Decreto 1720/2007, de 21 de diciembre.

Siguiendo lo dictado en el Artículo 81.3 del RD 1720/2007, se aplican las medidas de seguridad de nivel básico, medio y alto al referirse a información de salud de los individuos.

Con el fin de garantizar la disponibilidad de los datos, se realizarán copias de respaldo y recuperación como mínimo de semanalmente, salvo que en dicho período no se haya actualizado la información, según lo dispuesto en el Artículo 94.

El enfoque del sistema se considera de tipo *best effort* ya que se ha sido diseñado exclusivamente para cumplir con su cometido, sin realizarse la totalidad de pruebas y casos de estudio que garanticen la seguridad de la aplicación de forma certificada.

I. Aviso legal

Por favor lea la información contenida en los siguientes apartados con el fin de conocer las condiciones y términos de uso de la aplicación GeriApp. El usuario debe aceptar las presentes condiciones y términos de uso que tienen carácter vinculante, reservándose los desarrolladores el derecho a modificarlas.

Asimismo, debe comunicarle lo expuesto en este aviso legal al interno o tutor legal, que debe estar de acuerdo con su uso. Todo lo mencionado a partir de este punto es aplicable tanto a la aplicación móvil como al servidor.

II.Objeto

GeriApp es una aplicación diseñada para ofrecer una herramienta de gestión de la información personal y médica de los internos a los trabajadores de los centros geriátricos.

III. Garantía del servicio

La aplicación se proporciona "TAL CUAL", es decir, sin garantía, explícita o implícita, ni soporte de ningún tipo. Los desarrolladores de la aplicación se reservan el derecho de suspender el acceso a GeriApp sin previo aviso, por razones técnicas o de cualquier otra índole. Por ello, no se comprometen a ofrecer los servicios de la aplicación con ningún tipo de disponibilidad acordada.

IV. Limitación de responsabilidad

Tanto el usuario administrador como el personal de la residencia serán los únicos responsables de la utilización de la aplicación y de la toma de decisiones asociadas a la misma. Sin excepción alguna, no será posible culpar a los desarrolladores de la aplicación GeriApp de las consecuencias de su uso. De la misma manera, no se podrán reclamar daños morales o materiales, ni daños directos o indirectos, si la información ofrecida por la aplicación es inexacta o corrupta. Dado el nivel de desarrollo de la primera versión de la aplicación, no es posible evitar de manera fehaciente que usuarios no autorizados accedan a los datos tratados en la misma. En todo caso, la compensación máxima por el mal funcionamiento de la aplicación será de cincuenta euros, siempre y cuando se comunique en un período máximo de dos años.

Por todos los motivos y hechos mencionados en el presente apartado, se recomienda a los usuarios de la aplicación realizar un uso responsable y medido de la misma.

V. Política de privacidad

En cumplimiento de la Ley 15/99, de 13 de diciembre, de Protección de Datos de Carácter Personal, se comunica al usuario que la información tratada en la aplicación de GeriApp tiene como fin exclusivo prestar el servicio de gestión de la información personal y médica de los internos en un centro de mayores.

VI.Consideraciones fiscales

Todo el contenido y la funcionalidad de la aplicación tienen un propósito meramente informativo sin posibilidad de sustituir el juicio o las decisiones de un individuo cualificado. Así pues, la confianza en el sistema de recordatorios de GeriApp es por la cuenta y riesgo del usuario.

El uso de la aplicación móvil se rige por el ordenamiento jurídico español y por el contenido de este aviso legal. Por consiguiente, el acceso o el uso de la aplicación conllevan la aceptación de todas las condiciones expresadas anteriormente.

6 Gestión del proyecto

Este anexo recoge los aspectos más relevantes relacionados con el proyecto, tales como su planificación, los medios técnicos empleados durante su desarrollo y el coste asociado al mismo.

6.1 Planificación del proyecto

A continuación se expone la planificación inicial estimada para el proyecto y cuál ha sido el desarrollo real que se ha llevado a cabo.

6.1.1 Ciclo de vida seleccionado

Debido a la necesidad de un periodo de aprendizaje en algunas de las tecnologías a desarrollar al inicio del proyecto, se busca reducir el riesgo en las fases más críticas con el fin de acabar el sistema dentro de los plazos previstos. Así pues, el ciclo de vida seleccionado es el modelo de cascada con reducción de riesgos, que aúna las ventajas del modelo de cascada con las del modelo en espiral.

El proyecto completo se divide en tres sub-proyectos, cada uno enfocado a una tecnología diferente:

- Diseño y creación de la base de datos.
- Diseño y desarrollo de la aplicación móvil.
- Comunicación entre cliente y servidor.

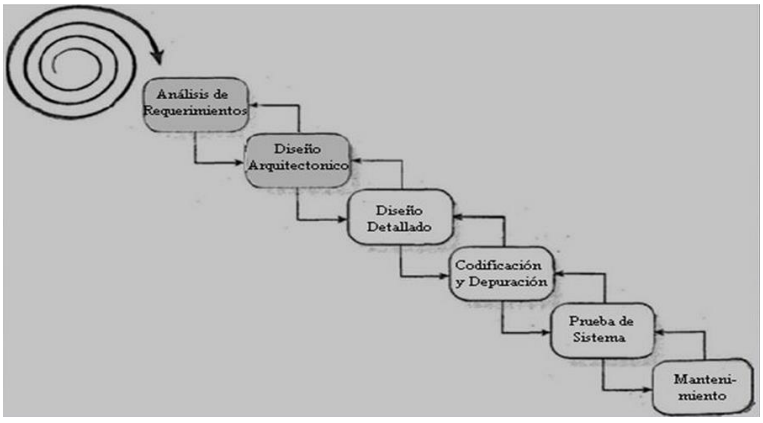


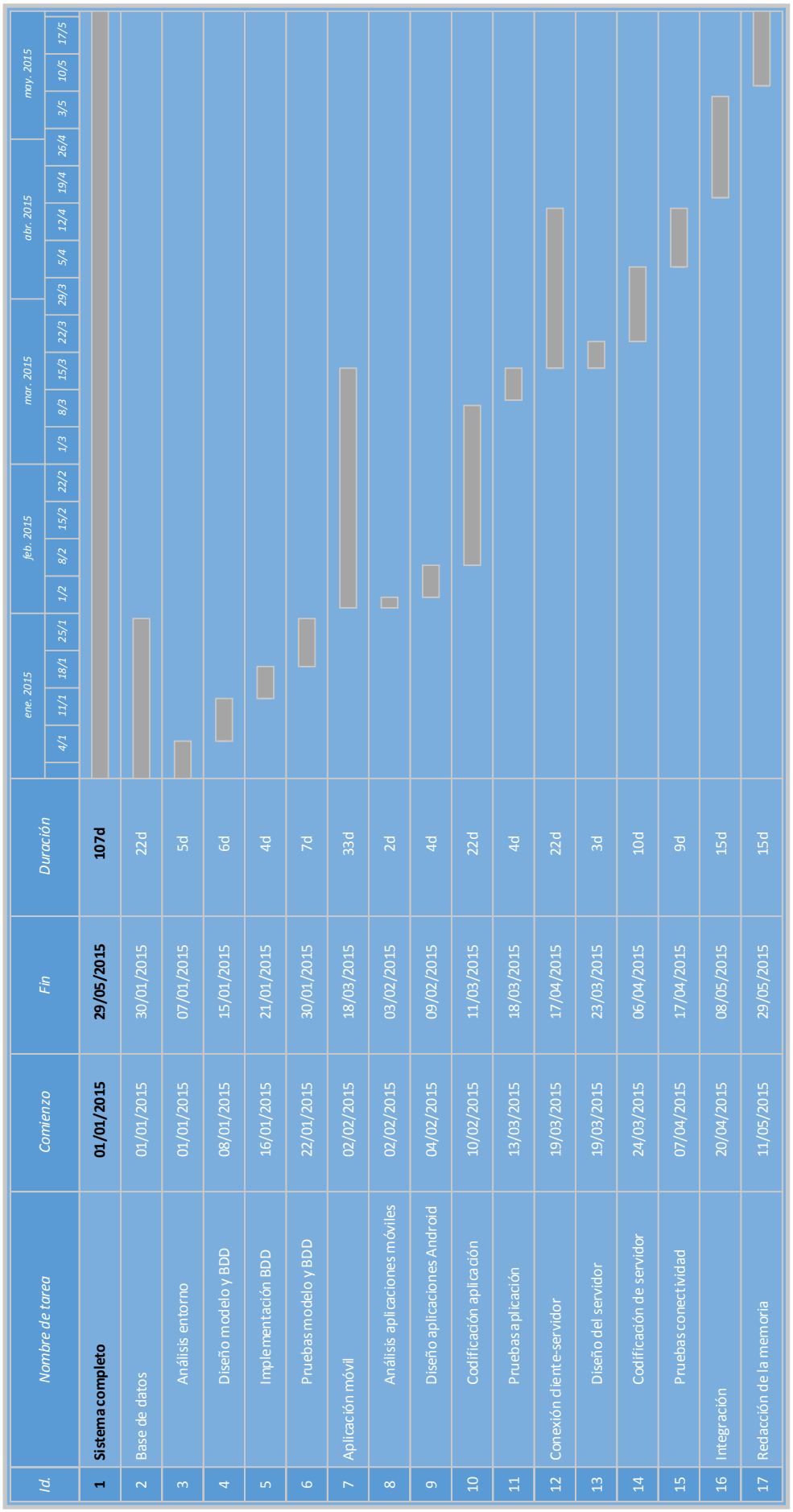
Figura 28: Modelo en cascada con reducción de riesgos
http://slideplayer.es/slide/21950

- **Inicio**. Desde un primer momento se busca satisfacer una necesidad potencial de un colectivo concreto como es el de los trabadores sanitarios de los centros geriátricos. Se define de manera genérica la idea a llevar a cabo y se estudia su viabilidad técnica.

- **Análisis**. Se analizan de manera más profunda las tecnologías aplicables a cada componente. Tras una serie de reuniones con el tutor del proyecto, se seleccionan las más adecuadas de acuerdo con los usos que se quieren satisfacer. Asimismo, se elabora una arquitectura preliminar del sistema y se identifican los requisitos de cada uno de los componentes.
- **Diseño**. Se aborda el diseño global de la solución técnica planteada ajustándose a los requisitos. Así, los componentes se descomponen en módulos, para los cuales se definen sus funciones y los diagramas necesarios para ilustrarlas.
- **Codificación**. Se implementan los distintos módulos usando el lenguaje y el entorno de programación adecuado.
- **Pruebas e integración**. Se realizan una serie de pruebas diferenciadas para cada componente y se integran los tres para validar que el sistema final cumple con todos los requisitos especificados.

6.1.3 Desarrollo del proyecto

El desarrollo real del proyecto GeriApp se indica en el siguiente diagrama de Gantt. Cabe señalar que la duración indica la cantidad de días laborables que supone cada una de las tareas.



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	**Sistema completo**	**01/01/2015**	**29/05/2015**	**107d**
2	Base de datos	01/01/2015	30/01/2015	22d
3	Análisis entorno	01/01/2015	07/01/2015	5d
4	Diseño modelo y BDD	08/01/2015	15/01/2015	6d
5	Implementación BDD	16/01/2015	21/01/2015	4d
6	Pruebas modelo y BDD	22/01/2015	30/01/2015	7d
7	Aplicación móvil	02/02/2015	18/03/2015	33d
8	Análisis aplicaciones móviles	02/02/2015	03/02/2015	2d
9	Diseño aplicaciones Android	04/02/2015	09/02/2015	4d
10	Codificación aplicación	10/02/2015	11/03/2015	22d
11	Pruebas aplicación	13/03/2015	18/03/2015	4d
12	Conexión cliente-servidor	19/03/2015	17/04/2015	22d
13	Diseño del servidor	19/03/2015	23/03/2015	3d
14	Codificación de servidor	24/03/2015	06/04/2015	10d
15	Pruebas conectividad	07/04/2015	17/04/2015	9d
16	Integración	20/04/2015	08/05/2015	15d
17	Redacción de la memoria	11/05/2015	29/05/2015	15d

Figura 29: Diagrama de Gantt para el desarrollo real del proyecto

6.2 Medios técnicos

Los medios técnicos empleados a lo largo del desarrollo de GeriApp se indican en los posteriores apartados, diferenciando claramente entre hardware y software.

6.2.1 Hardware

Para las labores de programación, depuración y testeo se han usado principalmente dos equipos informáticos personales. Un ordenador portátil Dell XPS 14z con Windows 8.1 (x64), un procesador Intel Core i7-2640M @ 2.8GHz y 8 GB de RAM y un ordenador de sobremesa corriendo Windows 7 (x64), con un procesador Intel Core i5-2400 @ 3,10 GHz y 8 GB de RAM (que a su vez hace las funciones de servidor).

La tableta usada ha sido la HTC Nexus 9 con la versión Lollipop de Android 5.1.1 que cuenta con una pantalla de 8,9" con una resolución de 2.048 x 1536 pixeles (287ppp), una CPU NVIDIA Tegra K1 dual-core @ 2,5 GHz (x64), una GPU NVIDIA Kepler de 192 núcleos y 2 GB de RAM (además de Wi-FI y NFC).

6.2.2 Software

ENTORNO DE PROGRAMACIÓN

Se ha desarrollado el proyecto con tres IDEs diferenciados. Para el desarrollo de la aplicación Android se usó **Android Studio** (en su última versión) al estar apoyado por la propia creadora del SO y sufrir numerosas actualizaciones y revisiones.

Para el despliegue del servidor Apache-Tomcat se ha usado **Eclipse** al tratarse de un programa conocido previamente y con múltiples posibilidades para importar módulos y trabajar con plugins externos.

Para el modelado y creación de la base de datos el programa elegido ha sido **MySQL Workbench 6.3**, debido a su simplicidad y a las múltiples posibilidades de ingeniería directa que aporta.

Cabe mencionar que fue necesario el uso puntual de los siguientes programas:

- **Andyroid**. Para tener un emulador más fluido que el proporcionado por AVD.
- **Intel HAXM**. Para acelerar el funcionamiento del AVD.
- **Apache-Tomcat 8.0**. Para la configuración del servidor.
- **TortoiseSVN**. Como utilidad para poder trabajar desde varios equipos controlando en todo momento las versiones y respaldos necesarios.

LIBRERÍAS EXTERNAS

- **Gson 2.2.4** y **Json 1.0**. Para manejar la información recuperada de la base de datos en formato JSON.
- **Androidplot-core-0.6.1**. Para graficar en pantalla la evolución de las constantes.
- **MySQL Connector/J 5.1.35**. Incluye el driver necesario para las operaciones en el servidor relacionadas con la base de datos MySQL.
- **Ndeftools-1.2.3**. Para trabajar con el sensor NFC.
- **Android SDK tools**. Con base el API 14.

HERRAMIENTAS DE DIAGRAMADO

Los diagramas de componentes, de clases, de secuencia y casos de uso se han diseñado con el programa **Microsoft Visio**.

El diagrama relacional de la base de datos se llevó a cabo con el propio **MySQL Workbench**.

HERRAMIENTAS DE PLANIFICACIÓN

Se ha empleado la herramienta Microsoft Project para realizar los diagramas de Gantt de la planificación inicial y final del proyecto.

OTRAS HERRAMIENTAS

Para la escritura de la memoria se ha empleado la suite completa de Microsoft Office 2013.

El navegador empleado para realizar la documentación e investigación y para realizar las diferentes pruebas del servidor ha sido Google Chrome con los siguientes plugins: ARC Welder, para la previsualización del archivo apk resultante y JSON Viewer para comprobar la devolución del servidor.

6.3 Análisis económico del proyecto

En este apartado se indican los costes previstos al inicio del proyecto y, posteriormente, el coste real del mismo. Asimismo, se indican las estimaciones y los cálculos realizados para obtenerlos. Cabe destacar que los precios de los productos o servicios indicados se corresponden con el momento de la consulta en el sitio web oficial del distribuidor. Los precios se corresponden a valores consultados con fecha del 01/06/2015.

Desde un primer momento se tomó la decisión de usar en la medida de lo posible soluciones de software libre y gratuito, o bien versiones de evaluación o de distribuciones para estudiantes sin coste alguno.

Como soporte hardware se consideró el uso de un ordenador de sobremesa para desarrollar el sistema y de un dispositivo móvil para realizar las pruebas de la aplicación móvil. El precio del ordenador estimado (maquina propietaria de la UC3M y creada modularmente de forma incremental) es de 599€ que debido a que el coeficiente de amortización anual según la Agencia Tributaria Española es de un 25%, se obtiene que la amortización anual es de 149,75 €. Como la duración del proyecto se estimó en seis meses, el coste asociado es de 74,875 €.

Respecto a la tableta electrónica, según la tienda de Google su precio recomendado es de 399 €. Aplicando el mismo razonamiento del párrafo anterior, el coste imputable al proyecto es de 49,875 €.

El coste por hora de trabajo para un Graduado en Ingeniería en Tecnologías de Telecomunicación se estima en 17 €. En cuánto al tutor sus horas han sido contabilizadas cómo de programador senior, estimándose a 25€ la hora.

Los costes de material comprenden la totalidad de accesorios de oficina que se han empleado durante el transcurso del proyecto, como folios, calculadora, etc… y además las etiquetas NFC.

Los costes de desplazamiento incluyen los viajes a Leganés para celebrar las reuniones con el tutor. Se han calculado teniendo en cuenta la media de consumos y distancias del vehículo propio (así como su desgaste), de este modo de ha calculado el coste medio diarios en 3€. Como se estimaba una reunión cada semana, veintidós en total, se deduce un coste de desplazamiento de 66 € en total.

Por último, los costes indirectos se estiman como un 10% del resto de costes y comprenden los gastos de electricidad y agua del personal, así como la tarifa de Internet consumida durante los seis meses del proyecto estimados.

6.3.1 Coste del proyecto

Los costes de personal, como se han dedicado unas 4 horas de media durante los 115 días del proyecto, se obtiene un coste de 7.820 €. En la siguiente tabla se muestran los cálculos del salario del personal.

Personal	Precio hora (€)	Horas/día	Días	Importe (€)
Programador Junior	17	4	115	7.820
Programador Senior	25	2	20	1.000
Total (con IVA)				**8.820**

Tabla 28: Desglose salario del personal

Las reuniones con el tutor se realizaron cada semana, pero los desplazamientos fueron mucho más numerosos al tener que trabajar en las dependencias del servidor. Se estima en 60 días los que se asistió a Leganés, acumulando un total de 180€ en la contabilidad de gastos.

Dado el desarrollo real del proyecto indicado anteriormente, la duración fue de cinco meses. Por ello, los costes finales han disminuido frente a los previstos, tal y como se observa en la siguiente tabla:

Concepto	Importe (€)
Licencias de software	0
Hardware	124,75
Personal	8.820
Material	50
Desplazamientos	180
Costes indirectos	817,48
Total (con IVA)	**9.992,23**

Tabla 29: Coste final del proyecto completo

Estas cantidades se han calculado de la misma manera que en la sección anterior, pero teniendo en cuenta que la duración del proyecto ha sido de cinco meses o 115 días.

6.3.3 Presupuesto para el cliente

Durante el primer año de la implantación del proyecto, se preveen numerosas modificaciones con el fin de resolver posibles problemas y mejorar las características actuales del sistema. Esto se computa como un 20% del total de costes del proyecto.

Es importante incluir un porcentaje de riesgo (15%), ya que la tecnología era desconocida al inicio del proyecto y por el posible desconocimiento del estado real del sector. En la tabla siguiente se recogen estos datos y, así, se puede consultar el coste para el proyecto completo:

Concepto	Importe (€)
Costes del proyecto	9.992,22
Costes de mantenimiento	1.998,45
Riesgos	1.498,83
Total (con IVA)	**13.489,50**

Tabla 30: Presupuesto para el cliente final

6.3.3 Beneficios

Partiendo del presupuesto estimado para el cliente, en este apartado se analizan las posibilidades para obtener beneficios económicos con el proyecto GeriApp. Así pues, se pueden destacar dos alternativas principales:

- Ofrecer el software ya preinstalado en dispositivos compatibles que se venderían con un sobrecoste y cobrar una cuota mensual en concepto

de mantenimiento y mejora tanto de la aplicación como de la base de datos.

- Ofrecer el software a medida del cliente, siendo éste quien proporcione los equipos (tantos dispositivos móviles como servidor) y encargarse de la instalación y configuración por un precio previamente acordado, siendo el cliente el responsable del mantenimiento de los equipos.

Ambas alternativas parten de la base de que se precisa el trato directo con el cliente a la hora de presentar/vender el software y que al tratarse de un nicho de mercado muy concreto no es conveniente publicar la aplicación en un ninguno de los mercados de aplicaciones existentes.

7 Conclusions and future work

This section describes the project results, the features that can be added to the system in the future and also describes the various ways in which it can be improved.

In addition, it explains what type of knowledge and skills acquired in the career are applied in the whole process of development of GeriApp.

7.1 Conclusions about the project

From the first time it was established the goal of getting a simple system using a mobile application. This system allows workers of any geriatric to have a tool to facilitate their work and especially to influence the benefit of the conditions and treatment of inmates in older seniors centers.

To achieve this end, the project was divided into three functional blocks: The database server that connects this with the application, the connection and the mobile application which acts as a client.

While the project itself has required a lot of time, dedication, foreground and added experience the fact of having a final product itself compensates all.

Another aspect that has surprised me on previous research, has been the willingness and initial acceptance of geriatric workers when advised and given their point of view of the needs the system should cover.

Previous interviews with nurses and geriatric workers were made to achieve an optimal solution. The principal aim has been to get a true reality system in which the data fields and values of the vital signs are real as possible.

Thus, we have obtained very close to the expectations of future users application and responds to their requests and tastes.

With a little more work and a series of demonstrations on site and testing periods, the system is feasible for installation in any kind of geriatric and established as an everyday tool.

It also has a high degree of evolution and improvement because it can easily adapt to new requirements and situations. It would be very simple to convert it into a similar system to manage hospitals, day care centers or disabled patients.

7.1.1 Project difficulty

Each of the modules has meant so much effort to us: learning to solve problems and seek solutions in different areas. This is due to the use of multiple technologies, platforms and languages and the need to find an optimal way to interconnect them.

One of the greatest difficulties arose when dealing with programming languages that are not handled with the ease which Java have been used to study many subjects throughout the race. The language that has been a challenge for me because its novelty was SQL, since it had not been used continuously during my training as an engineer.

Another difficulty has been the adaptation of the knowledge lifecycle and Java concurrency model to the specific use of Android and Activities for it.

There have been several technologies that have been learned in a deeper way, but certainly the most complex, in my view, is the model of connection based on JSON objects server, since he did not possess practical knowledge on the subject.

7.1.2 Project development

The project has been collated with working life dedicating the time necessary to obtain a satisfactory result within the time planned.

Of the three modules implemented, the one that took longest work and research were those encoding entailed Android application and the connection between tablet and server. The server module (both database and Apache-Tomcat) was much more intuitive and fluid took less technical additional tasks in case of database or more derived follow a procedure as in the case of the server.

The learning curve of the different IDEs, some of which I had not worked with in the past, has been very fast since it is intuitive software and adapted to the scope of the project. Despite this there have been numerous consultations to FAQs, wikis and forums in treated subjects.

Throughout the entire project has it followed a modular approach to move forward on a proven performance milestones and I have used several times a small test project in which to run code and test new structures and functions of the form quickly and safely. This is the case of JSON structures that were tested in an Eclipse project that was properly formed part to generate and test files that were interpreted by publishers JSON text mode.

7.1.3 Achieved result

The Project has achieved quite a goal result. The mobile application is designed following the guidelines of Android and its use is intuitive and simple. Moreover, it is not necessary to have computer skills for full use.

The application is only compatible with devices running Android 4.0 or higher and requires a device connected to the network all the time.

There where created a series of interfaces that allow us to manage tables of databases containing information of the nursing transparently.

GeriApp offers a complete system that could help people working in senior centers to perform their tasks and duties more effectively and easily.

The result consists of two separate modules, the client (tablets) and server (host), which are quite easy to install in any location and quick setup. GeriApp has been designed with the intention to distribute a small number of mobile devices per interconnected to the Wi-Fi network from the center ground and they are to be shared by nurses to make the rounds and then let them back to the control room for recharge.

One aspect that I think should be done to complete the project is a brief demonstration at some seniors center. This would provide a true view of system usage and would help to know which aspects should be improved or modified to obtain a better result.

7.2 Future work

Whenever a project of this magnitude is undertaken and as it is a system created by one person, with the help and guidance of the tutor, comes a compromise between time and details of the final result. For this reason there are some aspects, features or improvements that could not be added to the system but it is known that bring added value to it. Some of these things that have not decided to do are:

- **Database graphical management interface** that would provide the person responsible to sign in /sign out internal or modify sensitive information in a more usable tool to perform this task. This can be done with a simple web interface with a list of forms and editable text fields. This web is cross-platform and requires a previous step of authentication.

- **Alert** when a measurement is outside the normal range. The nurse is informed of a possible illness immediately. This will help detect diseases and abnormal health states of patients before they become serious.

- Android **notification service**. Using pop-up notifications on the taskbar could warn of the tasks that are close to expiring, thus recalling its performance without having to enter the list. Having visual and audible reminders will help to avoid forgetfulness and loss of activities.

- We can choose how often a **task repeats** and add it. For tasks that apply to a certain periodicity of a certain date to another.

- Add tasks to a specific **group of users**. Such as token passing and committed management. Working groups can be created to distribute tasks and divide responsibilities.

- **Login**, to identify who has done every action and have control over the workers. This aspect is one of the most important to control the work of the staff and have knowledge of the actual functioning of geriatric.

- **Security**. Using more stringent security mechanisms to keep the information in the application up to date. It's necessary use an encrypted channel and advanced security mechanisms to prevent malicious attacks and deny access to information from outsiders.

- **Advanced** use of **NFC**. There are already on the market various measurement devices (scales, thermometers, blood pressure...) compatible with NFC and Bluetooth. The possible integration of communication with them opens a lot of possibilities and helps to simplify the work of nurses.

- Add **artificial intelligence**. By using predictive and decisión computer algorithms it can provide intelligence to the system. Thus the system could recommend what actions to take in case of abnormalities in any vital sign. Through a learning process could codify rules of treatment or medication recommended for users with certain symptoms or diseases.

These are some of the future work that we may apply to the Project. Now is a good basis for a commercial application with principles of economic viability and increasing functionality that would gain higher value.

I'm not sure if it is worth investing time and effort in finding an economic profitability of the system. But what is certain is that I have acquired some skills and knowledge on the various platforms used that can help me in my future career.

Besides learning about different aspects of programming of client-server systems I discovered a new world that allows me to create a variety of applications and is interesting and fun.

Glosario de términos

En este capítulo se incluyen los acrónimos empleados a lo largo de este documento. Para cada uno, se indica su significado en inglés y una breve descripción en castellano.

- **API** (Application Programming Interface). Ofrece un conjunto de procedimientos y métodos para que diferentes componentes de software puedan interaccionar entre sí.
- **APK** (Android Application Package). Formato de archivo usado para distribuir e instalar aplicaciones en Android.
- **AVD** (Android Virtual Device). Emulador Android que permite modelarlo como un dispositivo real definiendo sus características de software y hardware.
- **DBMS** (Database Management System). Sistema Gestor de Base de Datos, es la tecnología empleada para el almacenamiento y la recuperación de información.
- **GUI.** Interfaz gráfica de usuario. Proporciona un entorno visual sencillo que permite la comunicación con el sistema operativo de una máquina o computador.
- **HTTP** (HyperText Transfer Protocol). Protocolo basado en el modelo petición-respuesta de datos a través de Internet.
- **IDE** (Integration Development Environment). Entorno de programación. Programa informático que proporciona diversas funciones y facilidades a los desarrolladores de software.
- **Java.** Lenguaje de programación de propósito general, concurrente, orientado a objetos.
- **JSON** (JavaScript Object Notation). Formato de intercambio de información basado en una colección de pares clave-valor e interoperable entre múltiples plataformas.
- **MAC.** Identificador de 48 bits que corresponde de forma única a una tarjeta o dispositivo de red, se conoce también como dirección física.
- **NFC** (Near Field Communication). Tecnología de comunicación inalámbrica de corto alcance que permite el intercambio de datos entre dispositivos.
- **OS** (Operating System). (Sistema Operativo), es el encargado de crear un vínculo entre los recursos hardware, el usuario y las aplicaciones de un sistema.
- **PHP** (PHP HyperText Processor). Lenguaje de código abierto muy popular y adecuado especialmente para el desarrollo web.
- **SDK** (Software Development Kit). En este documento se habla en particular de Android SDK, que proporciona las APIs y herramientas de desarrollo imprescindibles para diseñar, compilar, ejecutar y depurar aplicaciones en dicho sistema operativo.

- **UML** (Unified Modeling Language). Lenguaje de modelado de sistemas de software empleado para diseñarlos, especificarlos, desplegarlos y documentarlos.
- **URL** (Uniform Resource Locator). Cadena de caracteres que identifica un recurso en Internet.
- **XML** (eXtensible Markup Language). Lenguaje de marcado formado por etiquetas y basado en la interoperabilidad entre diferentes sistemas plataformas.

Bibliografía y referencias

1. Software de gestión integral para centros geriátricos. Gerosalus. [En línea] [Citado el: 1 de Febrero de 2014.] http://www.gerosalus.com/DefaultSalus.aspx.

2. Programa informático para la gestión de residencias de mayores y centros de día. Resiges. [En línea] [Citado el: 1 de Febrero de 2014.] http://www.resiges.com.

3. ResiPlus, Software de Gestión para Residencias de Mayores. ResiPlus. [En línea] [Citado el: 1 de Febrero de 2014.] http://www.addinformatica.com.

4. Teléfonos con NFC. Lista de teléfonos con NFC. [En línea] [Citado el: 1 de Marzo de 2014.] http://www.etiquetas-nfc.es/telefonos-smartphones-moviles-con-nfc.

5. NFC Enabled Phones And Tablets. Europe's largest supplier of NFC tags and products. [En línea] [Citado el: 1 de Marzo de 2014.] http://rapidnfc.com/nfc_enabled_phones.

6. Gizmondo. [En línea] [Citado el: 1 de Febrero de 2014.] http://es.gizmodo.com/esta-villa-holandesa-esta-disenada-al-completo-para-gen-1526918960.

7. CREATE & SHARE WIREFRAME DESIGNS FOR ANDROID. Ninjamock. [En línea] [Citado el: 1 de Abril de 2014.] http://www.ninjamock.com.

8. Guía completa de Android. Google. [En línea] [Citado el: 1 de Abril de 2014.] http://developer.android.com/develop/index.html.

9. Su fuente de conocimiento farmacológico. Vademecum. [En línea] [Citado el: 1 de Marzo de 2014.] http://www.vademecum.es.

10. Consumo de medicamentos en los ancianos: Resultado de un estudio poblacional. Revista Española de Salud Pública. [Versión impresa ISSN 1135-5727] [Citado el: 1 de Febrero de 2014.]

11. Guías y consejos de programación para Android. Androideity. [En línea] [Citado el: 1 de Abril de 2014.] http://androideity.com/category/programacion.

12. Conectar Java con MySQL usando JDBC. Código Fantasma. [En línea] [Citado el: 1 de Mayo de 2014.] http://www.codigofantasma.com/blog/conectar-java-con-mysql-usando-jdbc.

13. Crear conexión a servidor en MySQL Workbench. Yayosyayo. [En línea] [Citado el: 1 de Febrero de 2014.] http://www.slideshare.net/yayosyayo/crear-conexion-a-servidor-en-mysql-workbench.

14. Foro especializado en aplicaciones Android y programación avanzada. StackOverflow. [En línea] [Citado el: 1 de Mayo de 2014.] http://stackoverflow.com.

15. Acceso a MySQL desde Android con Eclipse, Java y JDBC. AJPDSoft. [En línea] [Citado el: 1 de Marzo de 2014.] http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=656.

16. Android NFC Resources, Requirements, and Setup. NFC Programming in Android. [En línea] [Citado el: 1 de Abril de 2014.] http://www.developer.com/ws/android/nfc-programming-in-android.html.

17. Gestión de Residencias de la Tercera Edad. Geriges. [En línea] [Citado el: 1 de Febrero de 2014.] http://www.geriges.com.

18. Software Online para la Gestión Completa de Residencias de Ancianos y Centros de Día. Software para Gestión de Residencias Geriátricas. [En línea] [Citado el: 1 de Febrero de 2014.] http://www.gestionderesidencias.es.

19. MySQL. The world's most popular open source database. [En línea] Oracle Corporation and/or its affiliates. [Citado el: 1 de Febrero de 2014.] http://www.mysql.com.

20. Pérez Leal, Raquel. Ciclo de vida de los proyectos - Proyectos de Telecomunicaciones. Madrid: UC3M, 2014 [Citado el: 1 de Junio de 2014.]

21. A Uniform Resource Name (URN) Namespace for the Near Field Communication (NFC) Forum. RFC 4729. [En línea] [Citado el: 1 de Abril de 2014.]
http://www.rfc-base.org/rfc-4729.html

22. RFCs variadas. IETF. [En línea] [Citado el: 1 de Abril de 2014.]
https://tools.ietf.org.

23. Android – Guía de desarrollo de aplicaciones para smartphones y tabletas. Sebastien Perochon y Sylvain Hebuterne. [ISBN: 9782746092297] [Citado el: 1 de Abril de 2014.]

24. Programación de bases de datos con MySQL y PHP. Helma Spona. [ISBN 9788426714688] [Citado el: 1 de Febrero de 2014.]

ANEXO 1: MANUAL DE USUARIO

En este anexo se incluye un breve manual de usuario de la aplicación Android que se ha desarrollado a lo largo del proyecto. Se debe recordar que es necesario tener conexión a Internet para la mayoría de las funciones de la aplicación.

Al abrir la aplicación desde el *launcher* de Android, ya sea desde el box de apps o desde su acceso directo si lo hubiese se le presenta al usuario la siguiente pantalla:

Figura 30: Vista principal de la pantalla de inicio de la aplicación móvil

En ella se le presentan mediante iconos grandes, claros y de fácil acceso las 6 principales opciones que ofrece GeriApp.

Empezaremos por la última de ellas. Al seleccionar el icono de ACERCA DE… se abra la siguiente interfaz informativa:

Figura 31: Interfaz informativa de la aplicación

En ella se muestra la información de GeriApp y del creador. El usuario puede volver a la pantalla inicial seleccionando el icono de la casa o ejecutando la orden de volver desde el panel de su terminal Android.

La segunda de las opciones ofrecidas es la que se abre al seleccionar el icono de TAREAS PENDIENTES:

Figura 32: Lista de las tareas pendientes

En esta vista se presentan (de 8 en 8) las tareas que no se han realizado aún. Se puede avanzar/retroceder en la lista tanto con las flechas situadas en la parte inferior como mediante un gesto de deslizamiento lateral.

El usuario puede seleccionar varias de las tareas, marcando la casilla *check* habilitada para tal uso y cambiarlas de estado al seleccionar la opción de *Marcar hecho*. En ese momento se refresca la pantalla, desapareciendo las tareas hechas. Desde este Activity el usuario también puede comprobar las actividades que ya se hicieron en el pasado seleccionado la opción de *Ver Hechos*. Es entonces cuando se presenta la siguiente pantalla:

Figura 33: Lista de las tareas hechas

En dicha interfaz el usuario sólo puede consultar la lista y volver al inicio de la aplicación o regresar a la lista de tareas sin hacer, seleccionando el icno correspondiente.

De nuevo en el MainActivity, el usuario puede seleccionar la tercera opción disponible seleccionando el icono en forma de signo matemático de suma. Es entonces cuando se abre la pantalla que muestra la opción para añadir tareas. Esta interfaz dispone de un cómodo reloj *scrollabe* que permite elegir fácilmente la hora de caducidad de la tarea y tres campos de texto que el usuario debe rellenar antes de guardar la tarea en la base de datos seleccionando el icono del disquete para salvar.

Además se proporciona un icono para descartar el texto introducido o borrar los valores por defecto.

Figura 34: Menú para añadir una nueva tarea

Una de las opciones más importantes es la cuarta, se accede a ella al seleccionar el icono nombrado como BASE DE DATOS. Es entonces cuando se muestra un listado con la totalidad de los usuarios activos en la residencia. Desde dicho interfaz el personal del centro puede abrir cualquiera de las fichas personales de los internos con tal sólo seleccionar su nombre.

Figura 35: Lista de los usuarios registrados

En la barra lateral de opciones también se ofrece la posibilidad de acceder a la búsqueda de un determinado usuario de forma rápida. Al seleccionar esta opción se abre un sencillo menú con una lista desplegable que permite buscar tanto por nombre como por cama.

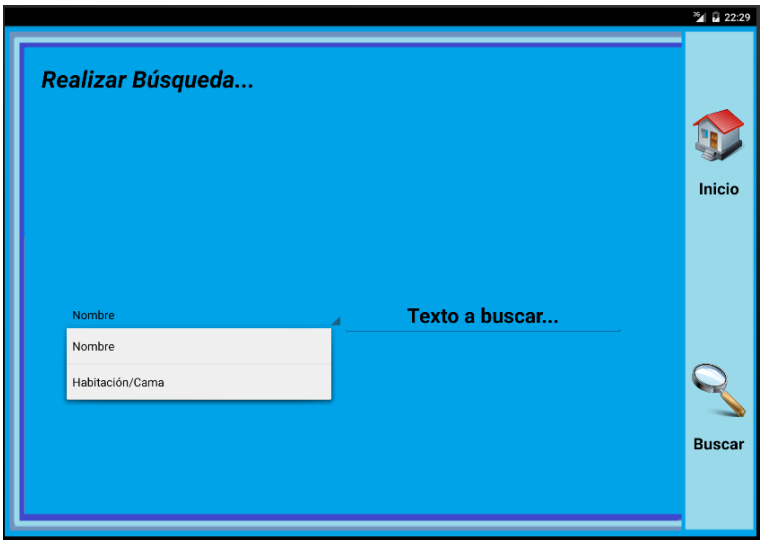


Figura 36: Menú de búsqueda de usuarios

Ya sea mediante una búsqueda, seleccionado el usuario de la lista o introduciendo el número de cama en la pantalla principal (quinta opción) se accede a la ficha personal del interno.

En esta vista se presentan los datos personales del mayor y un menú lateral con algunas opciones rápidas como son:

- Volver al inicio.
- Mostrar su foto.
- Abrir sus datos médicos.
- Ver la persona de contacto.

Figura 37: Vista de la ficha personal del interno

El icono de la cámara de fotos se encarga de presentar un pop-up con la foto de la persona que se ha consultado para su fácil identificación visual.

Figura 38: Pop-up con la fotografía del interno

El icono de contacto lanza el Activity encargado de mostrar el nombre de la persona de contacto y mostrar dos opciones para contactar con ella.

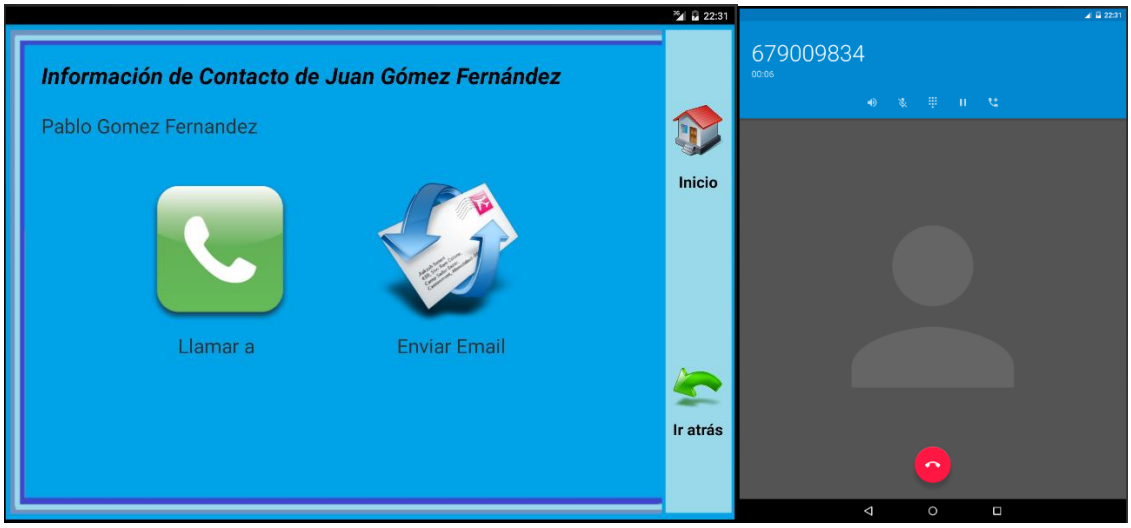


Figura 39: Vista de las opciones de comunicación y de la interfaz de llamada Android

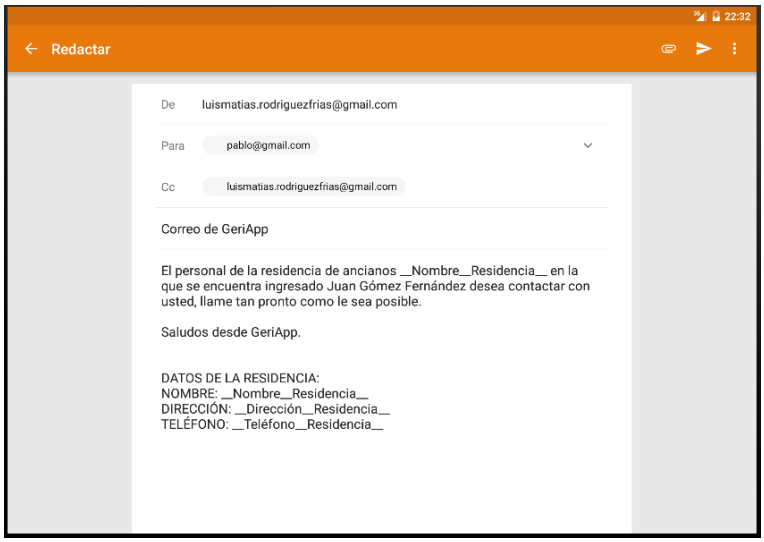


Figura 40: Captura del cuerpo del correo creado

La última de las opciones de la vista de la ficha personal es justo la que da acceso a la vista de la ficha médica. En ella se muestran los últimos valores de constantes tomados al interno y las enfermedades que sufre al igual que las medicinas que debe tomar.

Figura 41: Vista de la ficha médica del interno

Para acceder al historial cronológico del paciente es tan fácil como seleccionar el valor que se quiere consultar.

Figura 42: Vista del historial de una de las constantes

Este menú ofrece dos nuevas acciones. La de graficar los valores y la de añadir una nueva medida.

La primera de ellas muestra una lista de los valores posibles a mostrar en gráfica lineal. Se permite al usuario seleccionar sobre la lista qué consultar o ir moviéndose por la lista mediante las flechas.

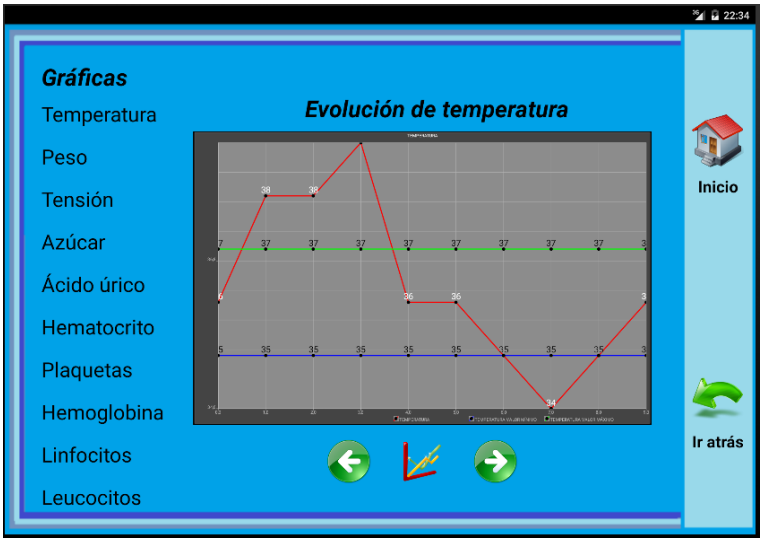


Figura 43: Demostración de las gráficas generadas

La otra opción que se presentaba en la ficha médica es la de añadir una medida. Esto se hace posible al poder editar los valores de los parámetros que se quiera.

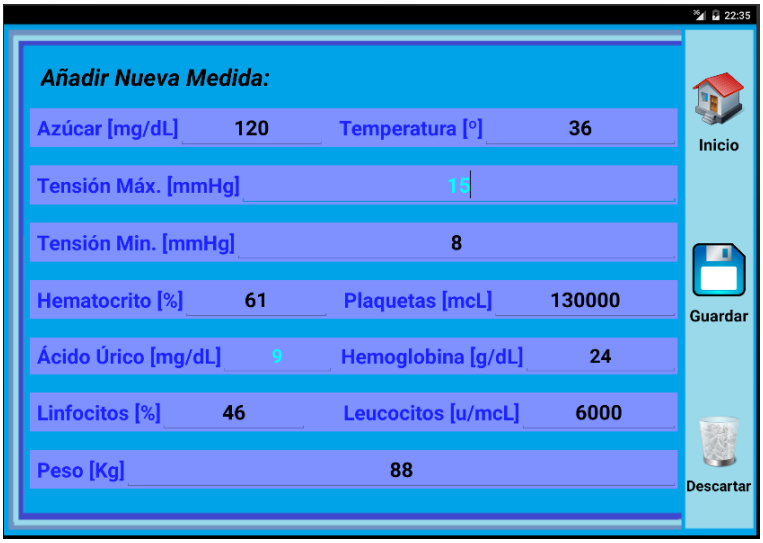


Figura 44: Menú para la adición de una nueva tanda de medidas

Un vídeo demostrativo se puede consultar en:

https://www.youtube.com/watch?v=BwyhKXK9HvM

ANEXO 2: MANUAL DEL PROGRAMADOR

En el presente anexo se detallan los pasos necesarios para disponer del entorno de desarrollo empleado por el autor de este proyecto. Para comenzar, se recomienda consultar la sección 6.2.2 en la que se indica el software imprescindible para poner en funcionamiento el sistema, en particular, las librerías y programas utilizados.

Las versiones del software adecuadas son las últimas a fecha de Junio del 2015 de cada uno de los programas.

A partir de este punto se asume que el usuario dispone de las siguientes soluciones de software:

- Android Studio, junto con las herramientas SDK.
- Eclipse IDE, preferiblemente la versión que integra la tecnología Java EE y el servidor web Apache Tomcat.
- MySQL Workbench.

Los siguientes apartados describen la secuencia de pasos a llevar a cabo para poner a punto la ejecución de la aplicación.

INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR MYSQL

Tras tener instalado el MySQL Workbench y un backup de la base de datos (tanto estructura como contenido) el usuario debe proceder de la siguiente forma para montar el servidor.

Una vez abierto el programa debe importar el modelo proporcionado, que le permitirá visualizar las tablas de la base de datos y relaciones que las definen.

Primero se ha de configurar la conexión de la siguiente forma:

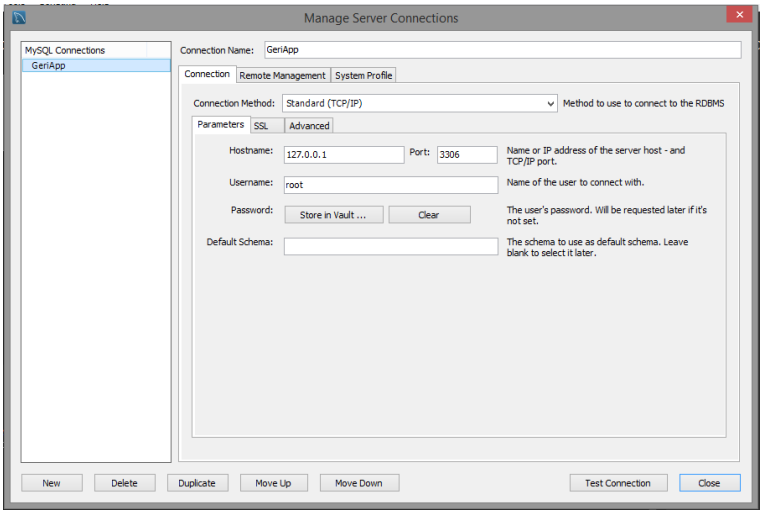


Figura 45: Configuración de la conexión de la base de datos

Es ahora cuando debe crear la base de datos seleccionando la opción *Database / Foward Engineer* de la IDE.

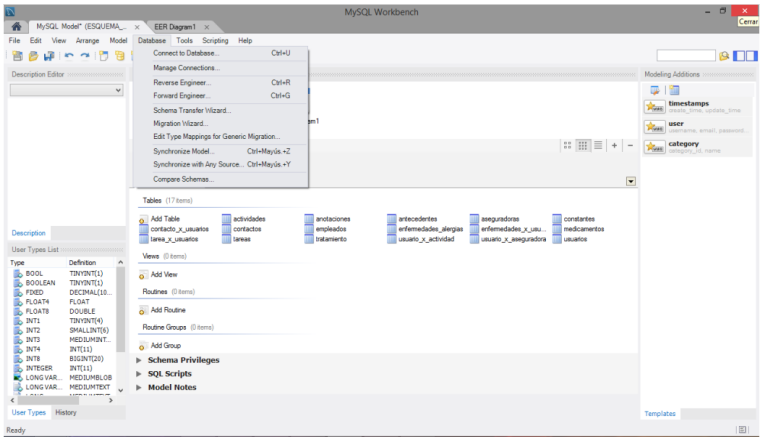


Figura 46: Menú para la creación y posterior sincronización de la base de datos en el servidor

Y posteriormente sincronizar el modelo con la opción correspondiente.

El usuario ya puede conectar con la base de datos (ctrl + U) y tras seleccionar la conexión a usar podrá probar la conectividad y el acceder al contenido de las tablas mediantes sentencias SQL.

Si se desea cargar una copia de seguridad de la base de datos se debe seleccionar la opción *Database / Data import* y posteriormente la opción de *Import Self Contained File*.

Para realizar consultas basta con introducir las sentencias SQL y seleccionar el icono del relámpago, resultando algo así:

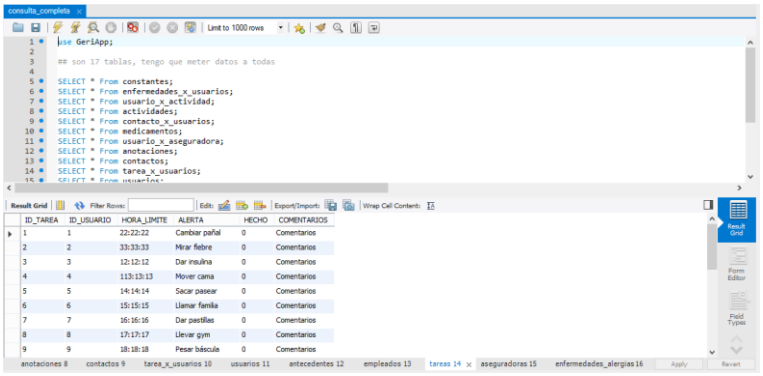


Figura 47: Ejemplo de la ejecución de sentencias MySQL de forma directa

Hay que recordar que cada modificación hecha en el modelo o en las tablas ha de ser verificada y aplicada mediante la opción de ingeniería directa y sincronización para que surta efecto.

PUESTA EN MARCHA DEL SERVIDOR APACHE-TOMCAT:

Una vez instalado y configurado Eclipse con los módulos de Java EE y de Tomcat. Se debe importar el proyecto proporcionado, dentro de este se encuentra la totalidad de código propio y librerías necesarias para este apartado.

Para lanzar el servicio que hará de servidor es necesario seleccionar la primera opción ofrecida al seleccionar el botón de play verde.

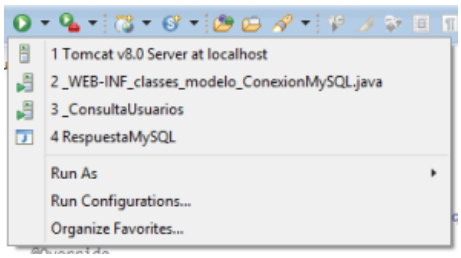


Figura 48: Menú para la ejecución del servidor tomcat

Esto hará que el servidor Apache comience a escuchar peticiones en el puerto pre configurado.

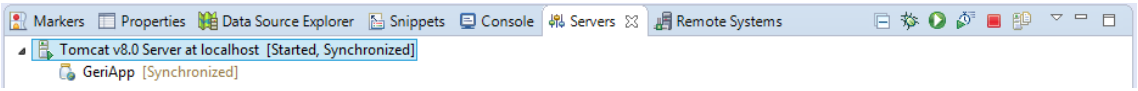


Figura 49: Vista del estado del servidor

La totalidad de parámetros de la conexión con la base de datos se pueden modificar fácilmente en el archivo *context.xml*.

MODIFICACIÓN DE LA APLICACIÓN ANDROID Y EMULACIÓN

El primero de los requisitos es tener instalado y configurado la última versión estable de Android Studio junto con el SDK. Una vez hecho esto se procede a la importación del proyecto proporcionado.

Para probar la app si no se dispone de una tableta de 9" o similar que cumpla los requisitos se puede configurar el AVD de la siguiente manera (es necesario saber que para este paso hace falta disponer un equipo con altos recursos).

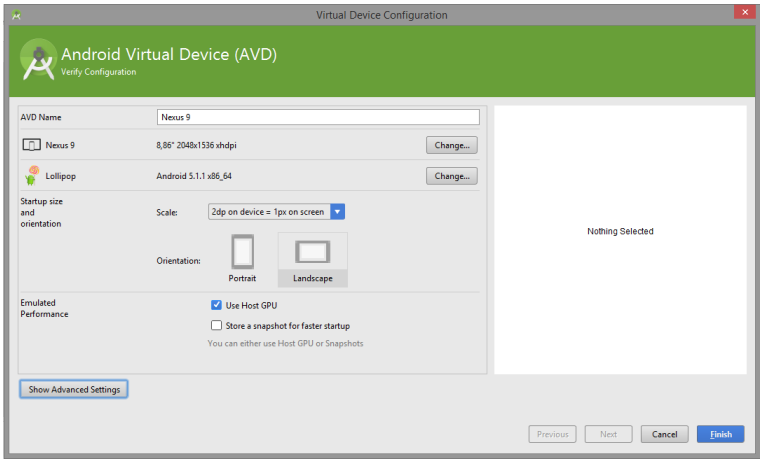


Figura 50: Configuración del AVD para la emulación de la tableta Nexus 9

Para visualizar el archivo de configuración usado por Shared Prefferences (preferencias.xml) es necesario usar la herramienta Android Device Monitor y acceder a la ruta:

Data/data/uc3m.geriappv2/shared_prefs/

Dicho archivo tiene el siguiente formato:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="DIRECCION">__Dirección__Residencia__</string>
    <string name="NOMBRE">__Nombre__Residencia__</string>
    <string name="TELEFONO">__Teléfono__Residencia__ </string>
</map>
```

La totalidad de variables de tipo texto incluidas se encuentran definidas en el archivo *strings.xml* y accediendo a él se pueden modificar sus valores.

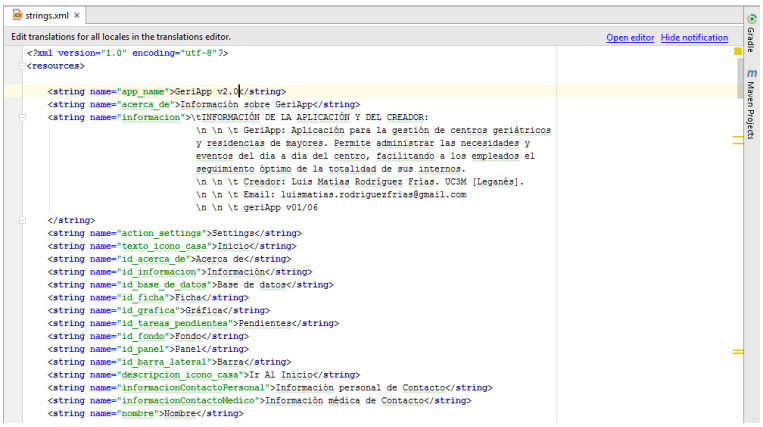


Figura 51: Demostración del contenido del archivo strings.xml

Se debe de conocer el ciclo de vida de las actividades en Android para poder añadir y/o modificar nuevas interfaces. En la siguiente figura se muestra en forma de diagrama el ciclo de vida de un Activity.

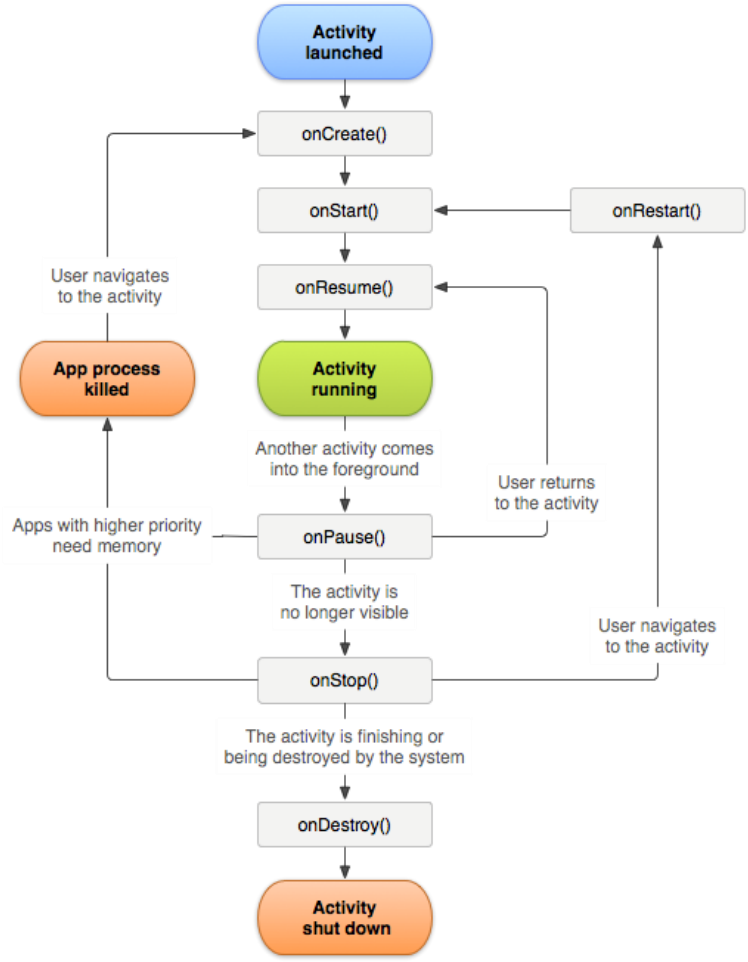


Figura 52: Ciclo de vida de un Activity Android

ANEXO 3: CONFIGURACIÓN DEL SERVIDOR APACHE TOMCAT

Se ha usado el servidor virtual de Eclipse para el desarrollo y pruebas de GeriApp, los archivos de configuración son los siguientes:

Context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xml>
<Context>
   <Resource   name="jdbc/db"
             auth="Container"
             type="javax.sql.DataSource"
             username="root"
             password="admin"
             driverClassName="com.mysql.jdbc.Driver"
             url="jdbc:mysql://163.117.141.185:3306/geriapp"
             removeAbandoned="true"
             removeAbandonedTimeout="1200"
             maxActive="20"
             maxIdle="10"
            maxWait="1000" />
</Context>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xml>
<resource-ref>
    <description> Datasource connection to db</description>
    <res-ref-name>jdbc/db</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
    <mapped-name>jdbc/db</mapped-name>
</resource-ref>
```

ANEXO 4: BASE DE DATOS MYSQL

consulta_completa

```
use GeriApp;

## son 17 tablas, tengo que meter datos a todas

SELECT * From constantes;
SELECT * From enfermedades_x_usuarios;
SELECT * From usuario_x_actividad;
SELECT * From actividades;
SELECT * From contacto_x_usuarios;
SELECT * From medicamentos;
SELECT * From usuario_x_aseguradora;
SELECT * From anotaciones;
SELECT * From contactos;
SELECT * From tarea_x_usuarios;
SELECT * From usuarios;
SELECT * From antecedentes;
SELECT * From empleados;
SELECT * From tareas;
SELECT * From aseguradoras;
SELECT * From enfermedades_alergias;
SELECT * From tratamiento;
```

Figura 53: Ejemplo de las consultas MySQL realizadas

test

```
use GeriApp;
SELECT * FROM usuarios
```

Result Grid

ID	DNI	NOMBRE	APELLIDOS	VALIDEZ	FECHA_NACIMIENTO	FECHA_INGRESO	DIETA	CAMA	FOTOGRAFIA	ACTIVO
1	53628725L	Juan	Gómez Fernández	Dependiente	1940-06-12	2012-12-12	Blanda	123	NULL	1
2	38158301Y	Pedro	López Rodríguez	Nula	1952-03-04	2012-12-12	Sin sal	456	NULL	1
3	18425846T	Antonio	Álvarez Gutierrez	Dependiente	1962-05-11	2012-12-12	Liquida	789	NULL	1
4	73928475H	David	Reina Mora	Independiente	1970-07-07	2012-12-12	No	124	NULL	1
5	38571946C	Sergio	García Jerez	Dependiente	1980-03-03	2012-12-12	Diabetica	125	NULL	1
6	10472636F	Jonás	Martín Sierra	Nula	1958-04-04	2012-12-12	Sin sal	126	NULL	1
7	19472947P	José	Prieto Costa	Dependiente	1932-07-08	2012-12-12	Blanda	347	NULL	1
8	31736203S	Ángel	Carreño Ruiz	Nula	1934-04-07	2012-12-12	No	367	NULL	1
9	28471746Z	Borja	Montero Espinosa	Dependiente	1931-01-02	2012-12-12	No	236	NULL	1
10	50118742W	Mario	Da Silva Rei	Independiente	1948-05-06	2012-12-12	No	558	NULL	1
11	28764537U	Juan	Ramírez García	Independiente	1960-09-09	2012-12-12	Sin sal	339	NULL	1
12	35228645Y	Ana	Marco Amor	Dependiente	1930-08-03	2012-12-12	Blanda	340	NULL	1
13	61946253R	Juana	Ruiz Castro	Independiente	1940-03-23	2012-12-12	Hipercal...	410	NULL	1
14	55227745T	Pascual	Huerta Quintana	Nula	1945-12-12	2012-12-12	No	412	NULL	1
15	48345678P	Miriam	Gutiérrez Sen	Nula	1940-03-23	2012-12-12	No	420	NULL	1

Figura 54: ejemplo de la devolución para una consulta

actividades - Table

Table Name: actividades Schema: GeriApp

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
CODIGO	INT(11)	✓	✓	✓				✓	
NOMBRE	VARCHAR(50)		✓	✓					
HORA_INICIO	INT(11)		✓						
HORA_FIN	INT(11)		✓						
LUGAR	VARCHAR(50)		✓						

anotaciones - Table

Table Name: anotaciones Schema: GeriApp

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
ID	INT(11)	✓	✓					✓	
USUARIOS_ID	INT(11)	✓	✓						
TIPO	VARCHAR(45)								
TEXTO	VARCHAR(45)								
FECHA	DATETIME								
EMPLEADOS_CODIGO_CREAD...	INT(11)	✓	✓						

Figura 55: Campos y tipos de las tablas "actividades" y "anotaciones"

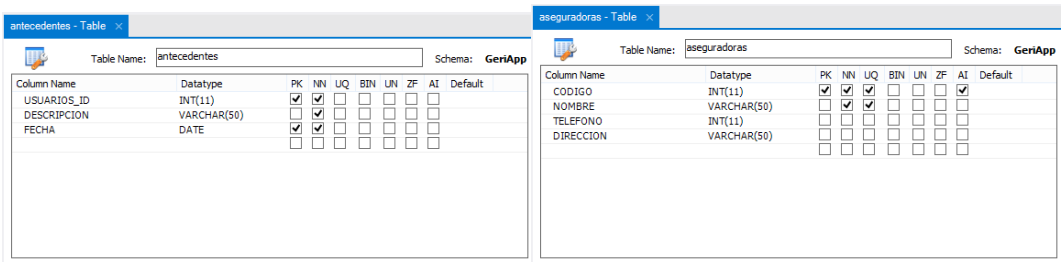


Figura 56: Campos y tipos de la tablas "antecedentes" y "aseguradoras"

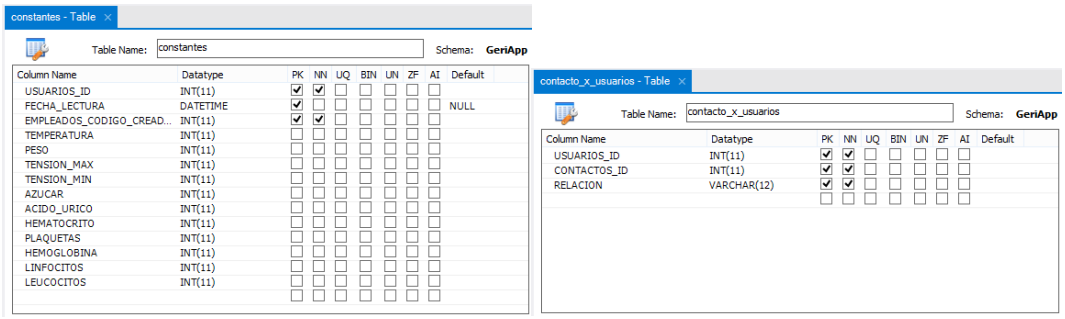


Figura 57: Campos y tipos de la tablas "constantes" y "contacto_x_usuarios"

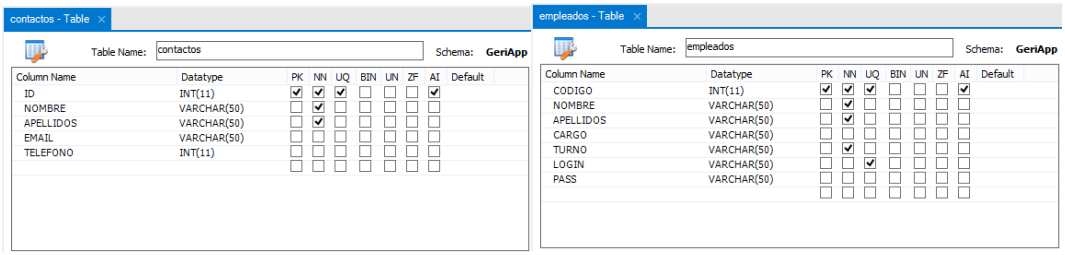


Figura 58: Campos y tipos de la tablas "contactos" y "empleados"

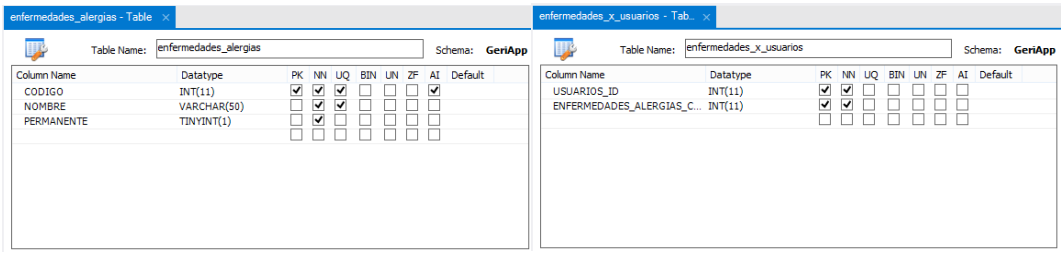


Figura 59: Campos y tipos de la tablas "enfermedades_alergias" y "enfermedades_x_usuarios"

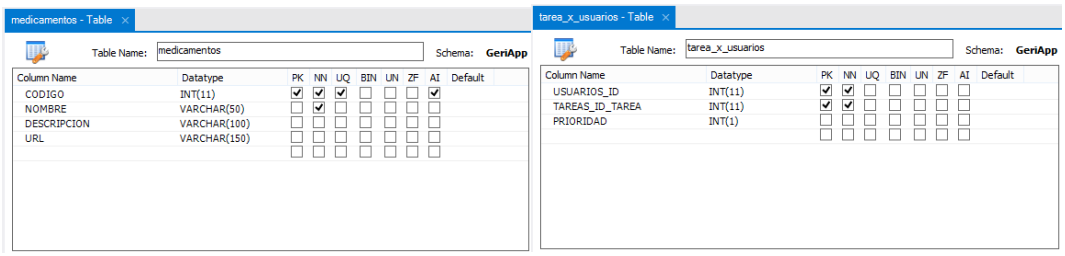


Figura 60: Campos y tipos de la tablas "medicamentos" y "tarea_x_usuarios"

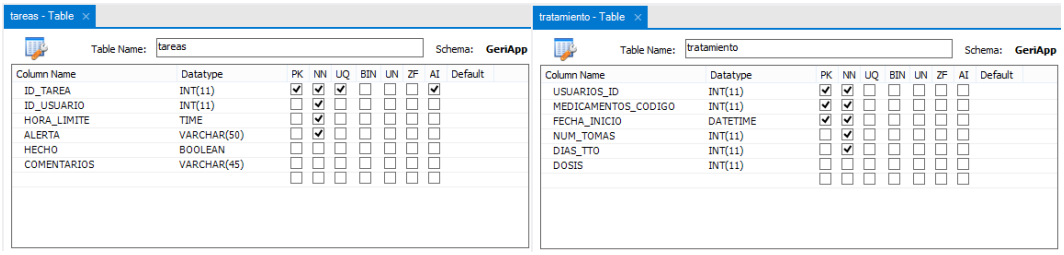


Figura 61: Campos y tipos de la tablas "tareas" y "tratamiento"

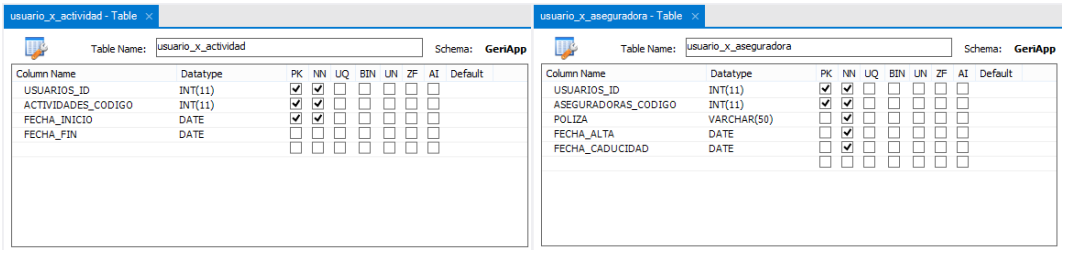


Figura 62: Campos y tipos de la tablas relacional "usuario_x_actividad" y "usuario_x_aseguradora"

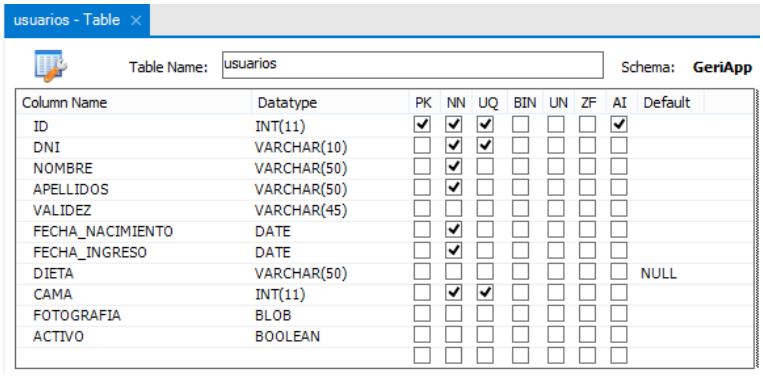


Figura 63: Campos y tipos de la tabla usuarios

ANEXO 5: PROYECTO ECLIPSE

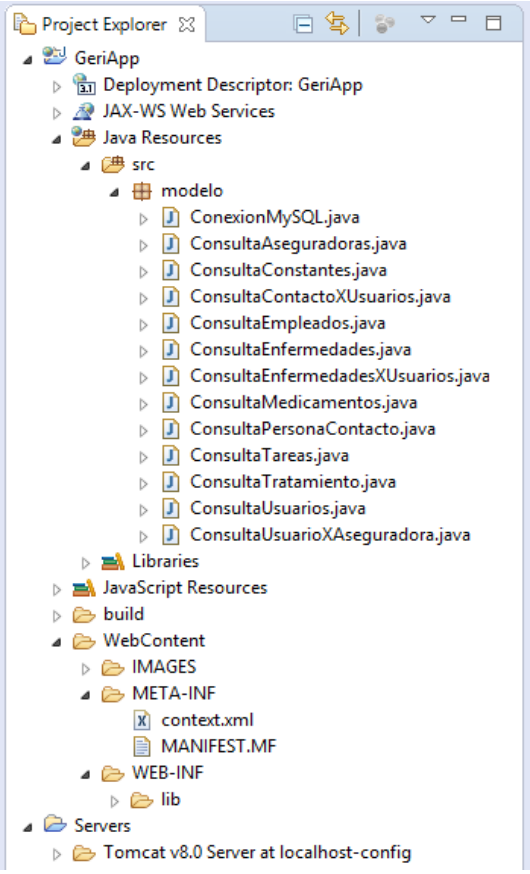


Figura 64: Estructura del proyecto completo en Eclipse para el servidor

ANEXO 6: ANDROID STUDIO

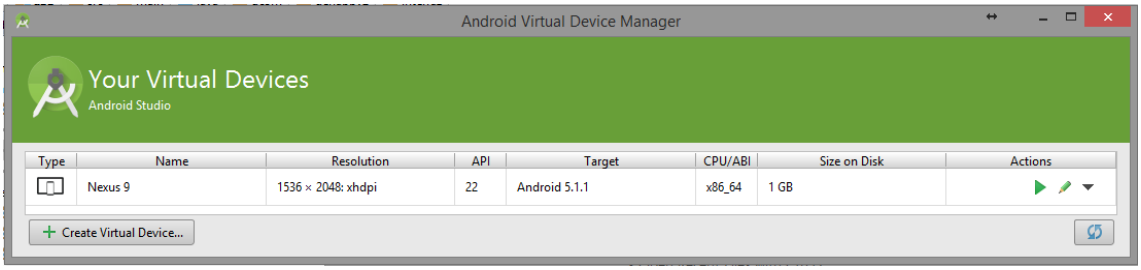


Figura 65: Vista del Android Virtual Device Manager

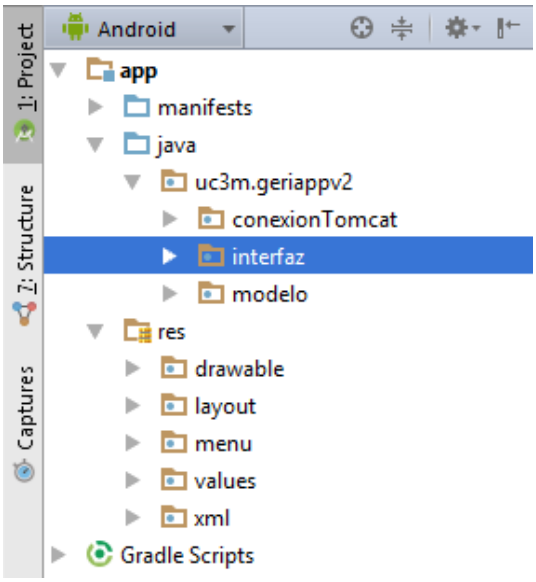


Figura 66: Estructura de los paquetes de la aplicación móvil

ANEXO 7: PERMISOS APLICACIÓN

Para el correcto funcionamiento de GeriApp se ha configurado la aplicación con permisos para usar las siguientes características:

```
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-feature
    android:name="android.hardware.nfc"
    android:required="false" />
<uses-feature
    android:name="android.hardware.telephony"
    android:required="false" />
```